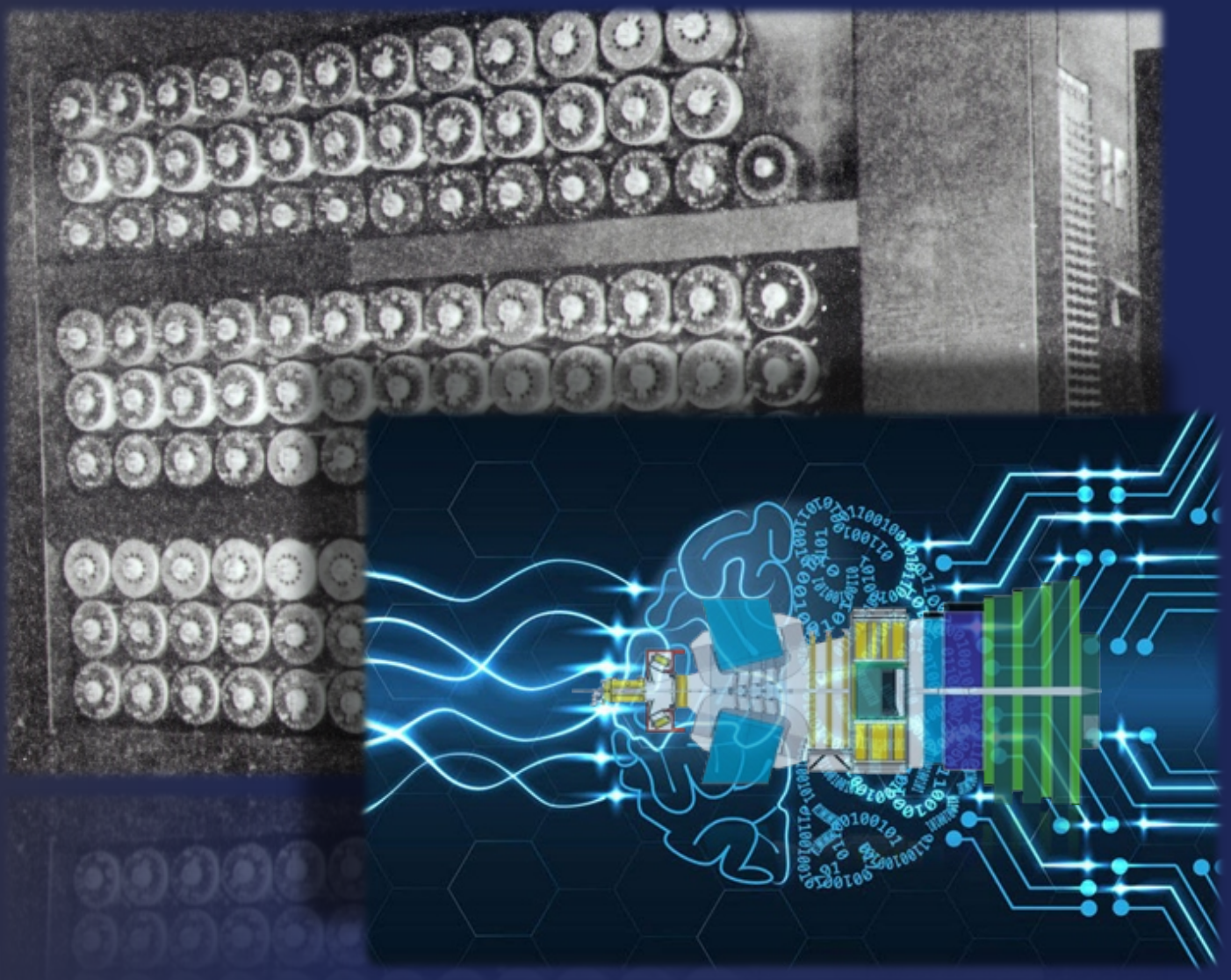
CERN/LHCC 2018-007
LHCb TDR 17
23 April 2018

UPGRADE

LHCb Software & Computing

TDR



Technical Design Report

LHCb Upgrade
Software & Computing
CERN/LHCC 2018-007

EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH (CERN)

CERN-LHCC-2018-007
LHCB-TDR-017
April 23, 2018

LHCb Upgrade Software and Computing Technical Design Report

The LHCb collaboration

Abstract

This document reports the Research and Development activities that are carried out in the software and computing domains in view of the upgrade of the LHCb experiment. The implementation of a full software trigger implies major changes in the core software framework, in the event data model, and in the reconstruction algorithms. The increase of the data volumes for both real and simulated datasets requires a corresponding scaling of the distributed computing infrastructure. An implementation plan in both domains is presented, together with a risk assessment analysis.

LHCb collaboration

I. Bediaga, M. Cruz Torres, J.M. De Miranda, A. Gomes[a], A. Massafferri, J. Molina Rodriguez[z], A.C. dos Reis, R. Santana, l. Soares Lavra, R. Tourinho Jadallah Aoude
[1] *Centro Brasileiro de Pesquisas Físicas (CBPF), Rio de Janeiro, Brazil*

S. Amato, K. Carvalho Akiba, F. Da Cunha Marinho, L. De Paula, F. Ferreira Rodrigues, M. Gandelman, A. Hicheur, J.H. Lopes, I. Nasteva, J.M. Otalora Goicochea, E. Polycarpo, C. Potterat, M.S. Rangel, L. Silva de Oliveira, B. Souza De Paula
[2] *Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil*

L. An, C. Chen, A. Davis, Y. Gan, Y. Gao, C. Gu, F. Jiang, T. Li, X. Liu, Z. Ren, J. Sun, Z. Tang, M. Wang, A. Xu, Z. Xu, Z. Yang, L. Zhang, W.C. Zhang[aa], X. Zhu
[3] *Center for High Energy Physics, Tsinghua University, Beijing, China*

M. Chefdeville, D. Decamp, Ph. Ghez, J.F. Marchand, M.-N. Minard, B. Pietrzyk, M. Reboud, S. T'Jampens, E. Tournefier, Z. Xu
[4] *Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, IN2P3-LAPP, Annecy, France*

Z. Ajaltouni, E. Cogneras, O. Deschamps, G. Gazzoni, C. Hadjivasiliou, M. Kozeiha, R. Lefèvre, J. Maratas[v], S. Monteil, P. Perret, B. Quintana, V. Tisserand, M. Vernet
[5] *Clermont Université, Université Blaise Pascal, CNRS/IN2P3, LPC, Clermont-Ferrand, France*

J. Arnau Romeu, E. Aslanides, J. Cogan, D. Gerstel, R. Le Gac, O. Leroy, G. Mancinelli, M. Martin, C. Meaux, A.B. Morris, J. Serrano, A. Tayduganov, A. Tsaregorodtsev
[6] *Aix Marseille Univ, CNRS/IN2P3, CPPM, Marseille, France*

Y. Amhis, V. Balagura[b], S. Barsuk, F. Bossu, D. Chamont, F. Desse, F. Fleuret[b], H. Grasland, J. Lefrançois, V. Lisovskyi, F. Machefert, C. Marin Benito, E. Maurice[b], V. Renaudin, P. Robbe, M.H. Schune, A. Usachov, M. Winn, G. Wormser, Y. Zhang
[7] *LAL, Univ. Paris-Sud, CNRS/IN2P3, Université Paris-Saclay, Orsay, France*

E. Ben-Haim, E. Bertholet, P. Billoir, M. Charles, L. Del Buono, G. Dujany, V.V. Gligorov, A. Mogini, F. Polci, R. Quagliani, F. Reiss, A. Robert, E.S. Sepulveda, D.Y. Tou, D. Vom Bruch
[8] *LPNHE, Université Pierre et Marie Curie, Université Paris Diderot, CNRS/IN2P3, Paris, France*

A. Khalfa
[9] *Centre de Calcul de l'Institut National de Physique Nucléaire et de Physique des Particules, Villeurbanne, France*

S. Beranek, M. Boubdir, S. Escher, A. Heister, T. Kirn, C. Langenbruch, M. Materok, S. Nieswand, S. Schael, E. Smith, T.A. Verlage, M. Whitehead, V. Zhukov[34]
[10] *I. Physikalisches Institut, RWTH Aachen University, Aachen, Germany*

J. Albrecht, A. Birnkraut, M. Demmer, U. Eitschberger, R. Ekelhof, L. Gavardi, K. Heinicke, P. Ibis, P. Mackowiak, F. Meier, A. Modden, T. Mombächer, J. Müller, V. Müller, R. Niet, S. Reichert, M. Schellenberg, T. Schmelzer, B. Spaan, H. Stevens, T. Tekampe, J. Wishahi
[11] *Fakultät Physik, Technische Universität Dortmund, Dortmund, Germany*

H.-P. Dembinski, T. Klimkovich, M.D.P. Peco Regales, M. Schmelling, M. Zavertyaev[c]
[12] *Max-Planck-Institut für Kernphysik (MPIK), Heidelberg, Germany*

P. d'Argent, S. Bachmann, D. Berninghoff, S. Braun, A. Comerma-Montells, M. Dziewiecki, D. Gerick, J.P. Grabowski, X. Han, S. Hansmann-Menzemer, M. Kecke, B. Khanji, M. Kolpin, R. Kopecna,

B. Leverington, J. Marks, D.S. Mitzel, S. Neubert, M. Neuner, A. Piucci, N. Skidmore, M. Stahl, S. Stemmle, U. Uwer, A. Zhelezov
[13] *Physikalisches Institut, Ruprecht-Karls-Universität Heidelberg, Heidelberg, Germany*

R. McNulty, N.V. Veronika
[14] *School of Physics, University College Dublin, Dublin, Ireland*

M. De Serio[d], R.A. Fini, A. Palano, A. Pastore, S. Simone[d]
[15] *INFN Sezione di Bari, Bari, Italy*

F. Betti[41], A. Carbone[e], A. Falabella, F. Ferrari, D. Galli[e], U. Marconi, D.P. O'Hanlon, C. Patrignani[e], M. Soares, V. Vagnoni, G. Valenti, S. Zucchelli
[16] *INFN Sezione di Bologna, Bologna, Italy*

M. Andreotti[g], W. Baldini, C. Bozzi[41], R. Calabrese[g], M. Corvo[g], M. Fiorini[g], E. Luppi[g], L. Minzoni[g], L.L. Pappalardo[g], B.G. Siddi, G. Tellarini, L. Tomassetti[g], S. Vecchi
[17] *INFN Sezione di Ferrara, Ferrara, Italy*

L. Anderlini, A. Bizzeti[u], G. Graziani, G. Passaleva[41], M. Veltri[r]
[18] *INFN Sezione di Firenze, Firenze, Italy*

P. Albicocco, G. Bencivenni, P. Campana, P. Ciambrone, P. De Simone, P. Di Nezza, S. Klaver, G. Lanfranchi, G. Morello, S. Ogilvy, M. Palutan[41], M. Poli Lener, M. Rotondo, M. Santimaria, A. Sarti[k], F. Sborzacchi, B. Sciascia
[19] *INFN Laboratori Nazionali di Frascati, Frascati, Italy*

R. Cardinale[h], G. Cavallero[h], F. Fontanelli[h], A. Petrolini[h]
[20] *INFN Sezione di Genova, Genova, Italy*

N. Belloli[i], M. Calvi[i], P. Carniti[i], L. Cassina, D. Fazzini[41,i], C. Gotti[i], C. Matteuzzi
[21] *INFN Sezione di Milano-Bicocca, Milano, Italy*

J. Fu[q], P. Gandini, D. Marangotto[q], A. Merli[q], N. Neri, M. Petruzzo[q]
[22] *INFN Sezione di Milano, Milano, Italy*

D. Brundu, A. Bursche, S. Cadeddu, A. Cardini, S. Chen, A. Contu, M. Fontana[41], P. Griffith, A. Lai, A. Loi, G. Manca[f], R. Oldeman[f], B. Saitta[f], C. Vacca[f]
[23] *INFN Sezione di Cagliari, Monserrato, Italy*

S. Amerio, A. Bertolin, S. Gallorini, D. Lucchesi[o], A. Gianelle, A. Lupato, E. Michielin, M. Morandin, L. Sestini, G. Simi[o]
[24] *INFN Sezione di Padova, Padova, Italy*

F. Bedeschi, R. Cenci[p], A. Lusiani, M.J. Morello[t], G. Punzi[p], M. Rama, S. Stracka[p], D. Tonelli, J. Walsh
[25] *INFN Sezione di Pisa, Pisa, Italy*

G. Carboni, L. Federici, E. Santovetti[j], A. Satta
[26] *INFN Sezione di Roma Tor Vergata, Roma, Italy*

V. Bocci, G. Martellotti, G. Penso, D. Pinci, R. Santacesaria, C. Satriano[s], A. Sciubba[k]
[27] *INFN Sezione di Roma La Sapienza, Roma, Italy*

J. Bhom, J. Brodzicka, A. Dziurda, W. Kucewicz[l], M. Kucharczyk, T. Lesiak, B. Malecki, A. Ossowska, M. Pikies, M. Witek, M. Zdybal
[28] *Henryk Niewodniczanski Institute of Nuclear Physics Polish Academy of Sciences, Kraków, Poland*

A. Dendek, M. Firlej, T. Fiutowski, M. Idzik, W. Krupa, M.W. Majewski, J. Moron, A. Oblakowska-Mucha, B. Rachwal, K. Swientek, T. Szumlak
[29] *AGH - University of Science and Technology, Faculty of Physics and Applied Computer Science, Kraków, Poland*

V. Batozskaya, H.K. Giemza, K. Klimaszewski, W. Krzemien, D. Melnychuk, A. Szabelski, A. Ukleja, W. Wislicki
[30] *National Center for Nuclear Research (NCBJ), Warsaw, Poland*

L. Cojocariu, A. Ene, L. Giubega, A. Grecu, T. Ivanoaica, F. Maciuc, V. Placinta, M. Straticiuc
[31] *Horia Hulubei National Institute of Physics and Nuclear Engineering, Bucharest-Magurele, Romania*

G. Alkhazov, N. Bondar, A. Chubykin, A. Dzyuba, K. Ivshin, S. Kotriakhova, O. Maev[41], D. Maisuzenko, N. Sagidova, Y. Shcheglov[†], M. Stepanova, A. Vorobyev
[32] *Petersburg Nuclear Physics Institute (PNPI), Gatchina, Russia*

F. Baryshnikov, I. Belyaev[41], A. Danilina, V. Egorychev, D. Golubkov, T. Kvaratskheliya[41], D. Pereima, D. Savrina[34], A. Semennikov
[33] *Institute of Theoretical and Experimental Physics (ITEP), Moscow, Russia*

A. Berezhnoy, I.V. Gorelov, A. Leflat, N. Nikitin, V. Volkov
[34] *Institute of Nuclear Physics, Moscow State University (SINP MSU), Moscow, Russia*

S. Filippov, E. Gushchin, L. Kravchuk
[35] *Institute for Nuclear Research of the Russian Academy of Sciences (INR RAS), Moscow, Russia*

K. Arzymatov, A. Baranov, M. Borisyak, V. Chekalina, D. Derkach, M. Hushchyn, N. Kazeev, E. Khairullin, F. Ratnikov[x], A. Rogozhnikov, A. Ustyuzhanin
[36] *Yandex School of Data Analysis, Moscow, Russia*

A. Bondar[w], S. Eidelman[w], P. Krokovny[w], V. Kudryavtsev[w], T. Maltsev[w], L. Shekhtman[w], V. Vorobyev[w]
[37] *Budker Institute of Nuclear Physics (SB RAS), Novosibirsk, Russia*

A. Artamonov, K. Belous, R. Dzhelyadin, Yu. Guz[41], V. Obraztsov, A. Popov, S. Poslavskii, V. Romanovskiy, M. Shapkin, O. Stenyakin, O. Yushchenko
[38] *Institute for High Energy Physics (IHEP), Protvino, Russia*

A. Alfonso Albero, M. Calvo Gomez[m], A. Camboni[m], J. Casals Hernandez, S. Coquereau, G. Fernandez, L. Garrido, D. Gascon, R. Graciani Diaz, E. Graugés, X. Vilasis-Cardona[m]
[39] *ICCUB, Universitat de Barcelona, Barcelona, Spain*

B. Adeva, A.A. Alves Jr, O. Boente Garcia, M. Borsato[41], V. Chobanova, X. Cid Vidal, A. Dosil Suárez, A. Fernandez Prieto, A. Gallas Torreira, B. Garcia Plana, M. Lucio Martinez, D. Martinez Santos, M. Plo Casasus, J. Prisciandaro, M. Ramos Pernas, A. Romero Vidal, J.J. Saborido Silva, B. Sanmartin Sedes, C. Santamarina Rios, P. Vazquez Regueiro, M. Vieites Diaz
[40] *Instituto Galego de Física de Altas Enerxías (IGFAE), Universidade de Santiago de Compostela, Santiago de Compostela, Spain*

F. Alessio, M.P. Blago, M. Brodski, J. Buytaert, W. Byczynski, D.H. Campora Perez, M. Cattaneo, Ph. Charpentier, S.-G. Chitic, M. Chrzaszcz, G. Ciezarek, M. Clemencic, J. Closier, V. Coco, P. Collins,

T. Colombo, G. Coombs, G. Corti, B. Couturier, C. D'Ambrosio, O. De Aguiar Francisco, K. De Bruyn, A. Di Canto, H. Dijkstra, F. Dordei, M. Dorigo[y], P. Durante, C. Färber, P. Fernandez Declara, M. Ferro-Luzzi, R. Forty, M. Frank, C. Frei, W. Funk, C. Gaspar, L.A. Granado Cardoso, L. Gruber, T. Gys, C. Haen, M. Hadji, C. Hasse, M. Hatch, B. Hegner, E. van Herwijnen, R. Jacobsson, D. Johnson, C. Joram, B. Jost, M. Karacson, D. Lacarrere, F. Lemaitre, R. Lindner, O. Lupton, M. Martinelli, R. Matev, Z. Mathe, D. Müller, N. Neufeld, A. Pearce, M. Pepe Altarelli, S. Perazzini, J. Pinzino, F. Pisani, S. Ponce, L.P. Promberger, M. Ravonel Salzgeber, M. Roehrken, S. Roiser, T. Ruf, H. Schindler, B. Schmidt, A. Schopper, R. Schwemmer, P. Seyfert, F. Stagni, S. Stahl, F. Teubert, E. Thomas, S. Tolk, A. Valassi, S. Valat, R. Vazquez Gomez, J.V. Viana Barbosa, B. Voneki, K. Wyllie
[41] *European Organization for Nuclear Research (CERN), Geneva, Switzerland*

G. Andreassi, V. Battista, A. Bay, V. Bellee, F. Blanc, M. De Cian, L. Ferreira Lopes, C. Fitzpatrick, S. Gianì, O.G. Girard, G. Haefeli, P.H. Hopchev, C. Khurewathanakul, A.K. Kuonen, V. Macko, M. Marinangeli, P. Marino, B. Maurin, T. Nakada, T. Nanut, T.D. Nguyen, C. Nguyen-Mau[n], P.R. Pais, L. Pescatore, G. Pietrzyk, F. Redi, A.B. Rodrigues, O. Schneider, M. Schubiger, P. Stefko, M.E. Stramaglia, M.T. Tran
[42] *Institute of Physics, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland*

C. Abellan Beteta, M. Atzeni, R. Bernet, C. Betancourt, Ia. Bezshyiko, A. Buonaura, J. García Pardiñas, E. Graverini, D. Lancierini, F. Lionetto, A. Mauri, K. Müller, P. Owen, A. Puig Navarro, N. Serra, R. Silva Coutinho, O. Steinkamp, B. Storaci, U. Straumann, A. Vollhardt, Z. Wang, A. Weiden
[43] *Physik-Institut, Universität Zürich, Zürich, Switzerland*

R. Aaij, S. Ali, F. Archilli, L.J. Bel, S. Benson, M. van Beuzekom, E. Dall'Occo, L. Dufour, S. Esen, M. Féo Pereira Rivello Carvalho, E. Govorkova, R. Greim, W. Hulsbergen, E. Jans, P. Koppenburg, M. Merk, M. Mulder, A. Pellegrino, C. Sanchez Gras, J. Templon, J. van Tilburg, N. Tuning[41], C. Vázquez Sierra, M. van Veghel, A. Vitkovskiy, J.A. de Vries
[44] *Nikhef National Institute for Subatomic Physics, Amsterdam, The Netherlands*

T. Ketel, G. Raven, V. Syropoulos
[45] *Nikhef National Institute for Subatomic Physics and VU University Amsterdam, Amsterdam, The Netherlands*

A. Dovbnya, S. Kandybei
[46] *NSC Kharkiv Institute of Physics and Technology (NSC KIPT), Kharkiv, Ukraine*

S. Koliiev, V. Pugatch
[47] *Institute for Nuclear Research of the National Academy of Sciences (KINR), Kyiv, Ukraine*

S. Bifani, R. Calladine, G. Chatzikonstantinidis, N. Farley, P. Ilten, C. Lazzeroni, A. Mazurov, J. Plews, D. Popov[12], A. Sergi[41], N.K. Watson, T. Williams, K.A. Zarebski
[48] *University of Birmingham, Birmingham, United Kingdom*

M. Adinolfi, S. Bhasin, E. Buchanan, M.G. Chapman, J. Dalseno, S.T. Harnew, J.M. Kariuki, S. Maddrell-Mander, P. Naik, K. Petridis, G.J. Pomery, E. Price, C. Prouve, J.H. Rademacker, S. Richards, J.J. Velthuis
[49] *H.H. Wills Physics Laboratory, University of Bristol, Bristol, United Kingdom*

M.O. Bettler, H.V. Cliff, B. Delaney, J. Garra Tico, V. Gibson, S.C. Haines, C.R. Jones, F. Keizer, M. Kenzie, G.H. Lovell, J.G. Smeaton, A. Trisovic, A. Tully, M. Vitti, D.R. Ward, I. Williams, S.A. Wotton
[50] *Cavendish Laboratory, University of Cambridge, Cambridge, United Kingdom*

J.J. Back, T. Blake, C.M. Costa Sobral, A. Crocombe, T. Gershon, M. Kreps, T. Latham, D. Loh, A. Mathad, E. Millard, A. Poluektov, J. Wicht
[51] *Department of Physics, University of Warwick, Coventry, United Kingdom*

S. Easo, R. Nandakumar, A. Papanestis, S. Ricciardi, F.F. Wilson[41]
[52] *STFC Rutherford Appleton Laboratory, Didcot, United Kingdom*

L. Carson, P.E.L. Clarke, G.A. Cowan, R. Currie, S. Eisenhardt, E. Gabriel, S. Gambetta, K. Gizdov, F. Muheim, M. Needham, M. Pappagallo, S. Petrucci, S. Playfer, I.T. Smith, J.B. Zonneveld
[53] *School of Physics and Astronomy, University of Edinburgh, Edinburgh, United Kingdom*

M. Alexander, J. Beddow, D. Bobulska, C.T. Dean, L. Douglas, L. Eklund, S. Karodia, I. Longstaff, M. Schiller, F.J.P. Soler, P. Spradlin, M. Traill
[54] *School of Physics and Astronomy, University of Glasgow, Glasgow, United Kingdom*

T.J.V. Bowcock, G. Casse, F. Dettori, K. Dreimanis, S. Farry, V. Franco Lima, T. Harrison, K. Hennessy, D. Hutchcroft, J.V. Mead, K. Rinnert, T. Shears, H.M. Wark, L.E. Yeomans
[55] *Oliver Lodge Laboratory, University of Liverpool, Liverpool, United Kingdom*

P. Alvarez Cartelle, S. Baker, U. Egede, A. Golutvin[70], M. Hecker, T. Humair, F. Kress, M. McCann[41], M. Patel, M. Smith, S. Stefkova, M.J. Tilley, D. Websdale
[56] *Imperial College London, London, United Kingdom*

R.B. Appleby, R.J. Barlow, W. Barter, S. Borghi[41], C. Burr, L. Capriotti, S. De Capua, D. Dutta, E. Gersabeck, M. Gersabeck, L. Grillo, R. Hidalgo Charman, M. Hilton, G. Lafferty, K. Maguire, A. McNab, D. Murray, C. Parkes, G. Sarpis, M.R.J. Williams
[57] *School of Physics and Astronomy, University of Manchester, Manchester, United Kingdom*

M. Bjørn, B.R. Gruberg Cazon, T. Hadavizadeh, T.H. Hancock, N. Harnew, D. Hill, J. Jalocha, M. John, N. Jurik, S. Malde, C.H. Murphy, A. Nandi, M. Pili, H. Pullen, A. Rollings, G. Veneziano, M. Vesterinen, G. Wilkinson
[58] *Department of Physics, University of Oxford, Oxford, United Kingdom*

T. Boettcher, D.C. Craik, C. Weisser, M. Williams
[59] *Massachusetts Institute of Technology, Cambridge, MA, United States*

S. Akar, T. Evans, Z.C. Huard, B. Meadows, E. Rodrigues, H.F. Schreiner, M.D. Sokoloff
[60] *University of Cincinnati, Cincinnati, OH, United States*

J.E. Andrews, B. Hamilton, A. Jawahery, W. Parker, J. Wimberley, Z. Yang
[61] *University of Maryland, College Park, MD, United States*

M. Artuso, B. Batsukh, A. Beiter, S. Blusk, S. Ely, M. Kelsey, K.E. Kim, Z. Li, X. Liang, R. Mountain, I. Polyakov, T. Skwarnicki, S. Stone, A. Venkateswaran, J. Wang, M. Wilkinson, Y. Yao, X. Yuan
[62] *Syracuse University, Syracuse, NY, United States*

C. Göbel, V. Salustino Guimaraes
[63] *Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil, associated to Institute* [2]

N. Beliy, J. He, W. Huang, P.-R. Li, X. Lyu, W. Qian, J. Qin, M. Saur, M. Szymanski, D. Vieira, Q. Xu, Y. Zheng
[64] *University of Chinese Academy of Sciences, Beijing, China, associated to Institute* [3]

L. Bian, H. Cai, L. Sun
[65] *School of Physics and Technology, Wuhan University, Wuhan, China, associated to Institute* [3]

B. Dey, W. Hu, Y. Wang, D. Xiao, Y. Xie, M. Xu, H. Yin, J. Yu[ab], D. Zhang
[66] *Institute of Particle Physics, Central China Normal University, Wuhan, Hubei, China, associated to Institute* [3]

D.A. Milanes, I.A. Monroy, J.A. Rodriguez Lopez
[67] *Departamento de Fisica , Universidad Nacional de Colombia, Bogota, Colombia, associated to Institute* [8]

O. Grünberg, M. Heß, N. Meinert, H. Viemann, R. Waldi
[68] *Institut für Physik, Universität Rostock, Rostock, Germany, associated to Institute* [13]

T. Likhomanenko, A. Malinin, O. Morgunova, A. Nogay, A. Petrov, A. Rogovskiy, V. Shevchenko
[69] *National Research Centre Kurchatov Institute, Moscow, Russia, associated to Institute* [33]

S. Didenko, N. Polukhina[c], E. Shmanin
[70] *National University of Science and Technology "MISIS", Moscow, Russia, associated to Institute* [33]

G. Panshin, S. Strokov, A. Vagner
[71] *National Research Tomsk Polytechnic University, Tomsk, Russia, associated to Institute* [33]

L.M. Garcia Martin, L. Henry, F. Martinez Vidal, A. Oyanguren, C. Remon Alepuz, J. Ruiz Vidal, C. Sanchez Mayordomo
[72] *Instituto de Fisica Corpuscular, Centro Mixto Universidad de Valencia - CSIC, Valencia, Spain, associated to Institute* [39]

C.J.G. Onderwater
[73] *Van Swinderen Institute, University of Groningen, Groningen, The Netherlands, associated to Institute* [44]

C.A. Aidala
[74] *University of Michigan, Ann Arbor, United States, associated to Institute* [62]

C.L. Da Silva, J.M. Durham
[75] *Los Alamos National Laboratory (LANL), Los Alamos, United States, associated to Institute* [62]

[a] *Universidade Federal do Triângulo Mineiro (UFTM), Uberaba-MG, Brazil*
[b] *Laboratoire Leprince-Ringuet, Palaiseau, France*
[c] *P.N. Lebedev Physical Institute, Russian Academy of Science (LPI RAS), Moscow, Russia*
[d] *Università di Bari, Bari, Italy*
[e] *Università di Bologna, Bologna, Italy*
[f] *Università di Cagliari, Cagliari, Italy*
[g] *Università di Ferrara, Ferrara, Italy*
[h] *Università di Genova, Genova, Italy*
[i] *Università di Milano Bicocca, Milano, Italy*
[j] *Università di Roma Tor Vergata, Roma, Italy*
[k] *Università di Roma La Sapienza, Roma, Italy*
[l] *AGH - University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Kraków, Poland*
[m] *LIFAELS, La Salle, Universitat Ramon Llull, Barcelona, Spain*
[n] *Hanoi University of Science, Hanoi, Vietnam*
[o] *Università di Padova, Padova, Italy*
[p] *Università di Pisa, Pisa, Italy*
[q] *Università degli Studi di Milano, Milano, Italy*
[r] *Università di Urbino, Urbino, Italy*

[s] *Università della Basilicata, Potenza, Italy*
[t] *Scuola Normale Superiore, Pisa, Italy*
[u] *Università di Modena e Reggio Emilia, Modena, Italy*
[v] *MSU - Iligan Institute of Technology (MSU-IIT), Iligan, Philippines*
[w] *Novosibirsk State University, Novosibirsk, Russia*
[x] *National Research University Higher School of Economics, Moscow, Russia*
[y] *Sezione INFN di Trieste, Trieste, Italy*
[z] *Escuela Agrícola Panamericana, San Antonio de Oriente, Honduras*
[aa] *School of Physics and Information Technology, Shaanxi Normal University (SNNU), Xi'an, China*
[ab] *Physics and Micro Electronic College, Hunan University, Changsha City, China*

[†] *Deceased*

Contents

Chapter 1

Overview and scope

The LHCb experiment will be upgraded for data taking in Run 3 and beyond [1]. The instantaneous luminosity will increase by a factor five, from 4×10^{32} to $2 \times 10^{33}\text{cm}^{-2}\text{s}^{-1}$. The lowest level trigger of the current experiment, a hardware-based trigger known as L0, will be removed. The L0 trigger has a hard limit of 1 MHz in its event output rate. A full software trigger will be deployed, with the goal of sustaining triggering capabilities for rates up to the inelastic proton-proton collision rate of 30 MHz. The processing and selection of events by software at this very high rate has a major impact on software and computing systems.

A study of the trigger output rates for different physics scenarios was reported earlier in the LHCb Trigger and Online Upgrade TDR [2]. In that document it was shown that

- a total event output rate of 100 kHz would allow to fully exploit both beauty and charm physics programmes of LHCb, while
- a diverse beauty programme and a charm programme of similar scope to that of Run 1 and Run 2 can be carried out with an event output rate of 50 kHz.
- an event output rate of 20 kHz would result in a restricted physics programme at LHCb.

There is therefore a strong motivation for recording data at an unprecedentedly high event rate. A strategy to cope with such high event rates has been developed over recent years, was put into place during the first long shutdown of the LHC (LS1) and has been used throughout Run 2. This strategy will be developed further for the upgrade as otherwise it would be impossible to record and fully process offline data at such a high rate. The main novel features introduced in Run 2 are:

- the High Level Trigger (HLT) was split into two stages [3], to enable the integration of a real-time calibration and alignment into the data taking process [4]; this, together with a better use of resources and an increased trigger farm size, allowed to run an online reconstruction equivalent to the offline one, thus enabling analyses to be performed on physics objects produced directly at the trigger level. A similar concept is foreseen for Run 3 where, however, the input rate of HLT will be 30 MHz instead of the current rate of 1 MHz, and the instantaneous luminosity will be a factor five higher;
- the `Turbo` data stream [5] was introduced to process a fraction of data (about 30% of the total event rate in Run 2). In this stream, the full raw data from detectors are no longer

stored but only a subset of the event information is saved, allowing to tailor the stored information to the specific needs of every single analysis. In 2017, the average event size of the `Turbo` stream was about half of the size of full raw events. This approach therefore considerably reduces the aggregate output bandwidth of the trigger and hence the required storage resources. Also, it does not require an offline reconstruction of the events, thus saving CPU resources.

The physics programme at the upgraded LHCb detector will be maximized by further exploiting these concepts, namely by fully moving event reconstruction and selection into the Event Filter Farm (EFF). The events accepted by the trigger will then be distributed as usual for offline analysis. In Run 3, the reconstruction and selection of events will take place at the trigger level and therefore it is expected that most, if not all, data will be processed by the `Turbo` stream. This implies that:

- the event format will be compact, and comparable to those currently used in Run 2 `Turbo` datasets (Micro Data Summary Tape, μDST), and
- since calibrations and alignments are performed on the EFF, there will be no need to re-reconstruct events offline; the offline data processing will thus be limited to streaming (with limited further selections, due to the higher purity of the selected data) and the storage costs for the recorded data will be driven essentially by the trigger output rate.
- the vast majority of offline CPU work will therefore be dedicated to the production of simulated events.

More details about the processing model in Run 2 and its natural evolution towards Run 3 are given in Chap. 2

A crucial ingredient in the upgrade programme is the ability of the event filter farm to process events at the LHC collision rate of 30 MHz. The size of the EFF for the upgrade was estimated in 2014 [2] using a growth-rate of server performance at equal cost that, although based on conservative estimates at that time, has turned out to be overly optimistic [6]. It is thus extremely important that the core framework architecture allows the available computing resources to be used as efficiently as possible. As shown in Fig. 1.1, the evolution of the trigger decisions per unit time in the existing software did not follow the expected increase of the computing power (as announced by vendors in GFlops) installed in the EFF, meaning that the LHCb software stack is not taking advantage of the full computing power offered by modern processors.

This is due to several factors, mostly to inefficient memory management (resulting in cache misses), excessive data manipulations, and inherently scalar computations (preventing the use of vector registers and/or of coprocessors such as GPGPUs and FPGAs). On the positive side, Fig. 1.1 shows that there is indeed much room for improvement, but this requires a paradigm shift in the core software framework, in the event data model and in the implementation of algorithms. One of the main challenges in the development of the upgraded EFF is that, while the CPU power of worker nodes increases significantly with time by increasing the number of cores, it becomes prohibitive to also increase the memory size accordingly. It is therefore crucial to be able to reduce the memory footprint, which facilitates an increase in the available CPU power at equal cost.

The design requirements, the R&D programme that is being performed, and the technical choices made in this area are reported in Chap. 3. The work reported in Chap. 3 provides a

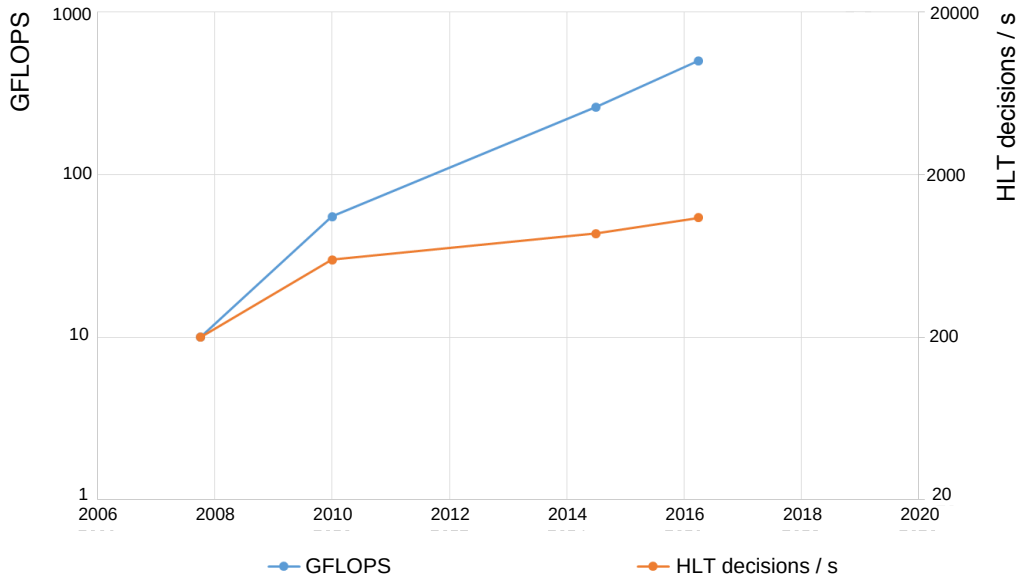


Figure 1.1: CPU performance in billion Floating Point Operations per second (GFLOPS) and trigger decisions per second for the 2014 version of the HLT, before the architecture upgrade reported in this document was implemented.

framework that enables to address some of the shortcomings mentioned above (*e.g.* the memory footprint per core), but the CPU shortcoming itself needs to be addressed mainly by algorithmic improvements (including the usage of new data structures).

An example of what can be achieved by implementing this strategy is reported in Chap. 4. The possibility of exploring other hardware architectures in addition to x86 CPUs in order to achieve the needed trigger performance is discussed in Chap. 5.

Since the production of Monte Carlo events will, even more than presently, dominate the offline CPU needs, it is mandatory to speed up the simulation applications, by implementing faster or parameterised simulations (to be used for quick analysis prototyping, background studies, and the estimation of part of systematic uncertainties) and by speeding up the full simulation based on GEANT4 [7–9] (to be used in any case to for determining signal efficiencies). Chap. 6 reports on the design and technical implementation for faster simulation applications.

The emergence of new paradigms of distributed computing (private and commercial clouds, High Performance Computing clusters, volunteer computing...) are already now changing the traditional landscape of the Grid computing used offline. It is therefore crucial to have a very versatile and flexible system for handling distributed computing (production and user data analysis). Chap. 7 details the current status and the challenges to be dealt with in this domain, specifically from the infrastructure point of view. Aspects more related to the computing model and the associated resource requirements for Run 3 are described in a separate document [10].

Dependencies on external software packages, software infrastructure and collaborative working tools are detailed in Chap. 8. Issues related to data and analysis preservation are discussed in Chap. 9. Project organization, planning and milestones are reported in Chap. 10, along with a risk register.

In all the following chapters, emphasis has been given to the changes with respect to the current software and computing, that are described in the 2005 LHCb Computing TDR [11] and in the Run 2 update of [12].

Chapter 2

Data processing model

The LHCb experiment covers a wide range of physics measurements, with focus on the study of charm and beauty decays. Charm and beauty hadrons are produced copiously in the proton-proton collisions at the LHC. This poses two challenges to the processing model in terms of output bandwidth: first, events have to be selected with a high purity and, second, the event information available to analysts needs to be reduced. All this is implemented in the LHCb data processing model from event triggering down to ntuple production for analysis. In the following sections the processing model in Run 2 and the necessary modifications for Run 3 are presented.

2.1 Online data processing model in Run 2

2.1.1 The Run 2 novel HLT paradigms

The trigger system of LHCb consists of two stages. The first stage, L0, is implemented in hardware and reduces the LHC collision rate to about 1 MHz[1] by using information from the calorimeter and muon systems. After L0 the full detector information is read out and events are sent to the Event Filter Farm, EFF, where the second stage, the software-based High Level trigger (HLT), reduces the rate further. The innovative and novel feature introduced since the beginning of Run 2 is the separation in two stages of the HLT. The first stage (HLT1), performs a partial event reconstruction, due to the tight time budget. The output of HLT1 is then recorded to the local hard disks of each node of the EFF for later asynchronous processing in the second software stage (HLT2). This strategy has two benefits that are crucial to the Run 2 and Run 3 processing model:

- it increases the effective resources of the EFF as HLT2 is fully asynchronous and events can therefore be processed at any time independently of the LHC activity; the implementation of the split HLT are discussed in more detail in Sec. 2.1.2;
- it allows to calibrate and align all sub-detectors with data collected by HLT1 in real-time before HLT2 starts processing these events. The real-time alignment and calibration system is described in Sec. 2.1.3.

As a consequence, HLT2 can perform a full event reconstruction, including the full track reconstruction and the full particle identification, with a performance similar or better than the

[1] This rate is limited by the design of the buffering hardware.

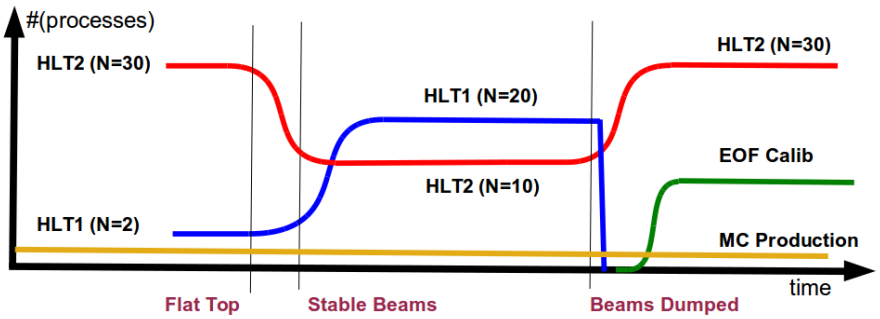


Figure 2.1: A schematic diagram showing the amount of CPU dedicated to the HLT1 and HLT2 activity during various states of the LHC.

offline reconstruction in Run 1, in terms of efficiency and resolution.

Both HLT1 and HLT2 are configured and run in the online environement by different configurations of the MOORE application (see Fig. 2.3).

As the full event information is available already in the trigger, LHCb has implemented the possibility to perform analyses directly on the trigger output information (called `Turbo` stream). In the `Turbo` stream, the raw event information is discarded and only a few trigger objects are persisted, which leads to a big reduction in event size and to a higher output rate for a fixed bandwidth. Performing the full event reconstruction is also beneficial for the selections saving the full event information, as they can be more efficient and more pure.

2.1.2 Implementation of HLT1 and HLT2 concurrency

The operation of the LHCb online system during Run 2 showed clearly that the optimized usage of the available CPU power on the Event Filter Farm highly depends on flexible and instantaneous tuning mechanisms for the concurrently running applications. In practice the main users of the HLT farm, HLT1 and HLT2 can very quickly adapt to changing CPU needs, which depend largely on the state of the LHC. An optimal working point was identified where an adjustable CPU load is generated by executing HLT2. The fraction of CPU dedicated to HLT2 rises to a higher level while LHC does not deliver stable physics conditions and, as sketched in Fig. 2.1, the amount of CPU dedicated to HLT2 is ramped down in order to free resources for the HLT1 activity as soon as the LHC collider starts to deliver physics collisions. Once data taking stops at the end of an LHC fill, HLT1 resources are reduced and HLT2 resources are ramped up to a level that still allows other necessary activities such as calibrations or standalone sub-detector tests. In addition, if allowed by the load, the EFF can reserve a small amount of CPU time for simulated event production, as shown in Fig. 2.1.

Presently this functionality is achieved by a fork-on-demand mechanism, where additional processing applications immediately starting to process events can be created with roughly no overhead and with minimal memory foot-print by a dedicated process forking mechanism. One important aspect of this scheme is that processes are forked, *i.e.* they do not require initialization time.

2.1.3 Real-time calibration and alignment

LHCb has introduced a real-time detector alignment and calibration strategy for LHC Run 2, and is planning to adopt the same very successful approach for Run 3. Data collected at the start of a fill are processed within a few minutes and used to update the detector alignment, while the calibration constants are evaluated for each run.[2] When the new set of detector conditions is significantly different from the one used to take the data, the value of the conditions is updated and the new constants are used in the trigger reconstruction.

This procedure is one of the key elements that allow to have offline quality reconstructed data already at trigger level, with no need for further processing. This is crucial for the data processing model adopted for Run 3 where no offline processing is foreseen.

The implementation of the real-time alignment and calibration in Run 3 will be similar to that used in Run 2, but the procedure of each alignment task will be studied and improved in order to match better the physics requirements and achieve better performance.

Most of the calibration tasks, like for example the refractive index calibration of the RICH detectors, are implemented as algorithms that analyse one or more monitoring histograms produced by the online reconstruction tasks, while the alignment of the detector, which includes alignment of the VELO, tracking system, mirrors of the RICH detectors and Muon system, is obtained by means of iterative procedures exploiting the parallel processing of the events in the Event Filter Farm (EFF) nodes. The calibration of the electromagnetic calorimeter is implemented in a similar fashion.

The new detector conditions calculated by the alignment and calibration tasks are then applied in HLT2. This means that the data passing the HLT1 selection need to remain on the buffers of the EFF nodes until the complete set of detector conditions is available. In addition, the conditions used in the HLT1 reconstruction are also updated with the results of alignment and calibrations as soon as they are available.

The main elements and requirements of the real-time alignment and calibration process can be summarised as follows:

- *Interaction with the Online framework, process steering and control via Finite State Machine:* all the transitions must be steered and information exchanged by means of network messages.
- *Buffering of the events:* the events selected by HLT1 need to be buffered before starting the HLT2 processing, for the time necessary to obtain the updated complete set of detector conditions. As an example, the usage of the disk buffer during the 2016 *pp* run is reported in Fig. 2.2
- *Alignment application:* an application must be provided that performs reconstruction on saved events passing the HLT1 selection and to calculate the new alignment constants.
- *Event model:* in order to obtain a high quality alignment, tracks may need to be refitted with respect to the simplified reconstruction performed in HLT1; the event model must thus allow refitting and computation of covariance matrices, while remaining compact.
- *Online monitoring:* it should be possible to analyse monitoring histograms, both during the calibration tasks execution and for monitoring the alignment and calibration results.

[2] A *run* corresponds to about one hour of data taking

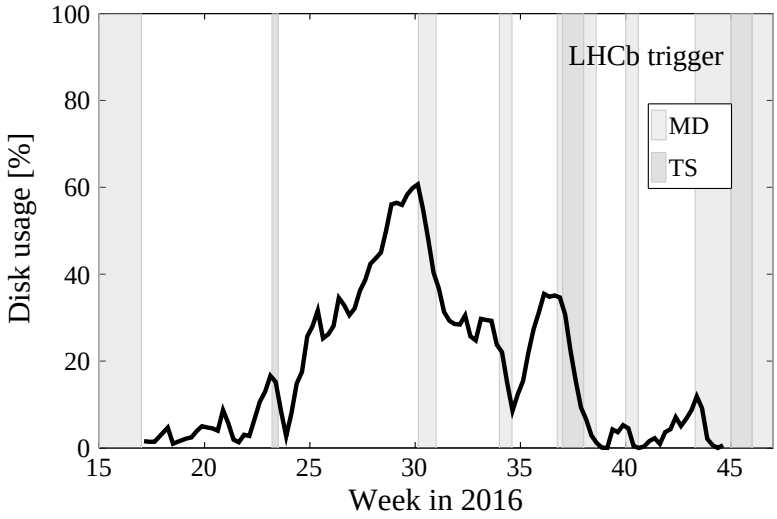


Figure 2.2: Disk buffer usage during the 2016 *pp* run. The HLT1 rate was adjusted 3 times based on projections under the assumption of a sustained machine efficiency of 70%. At week 24, the HLT1 output rate was reduced to from 130 to 115 kHz. In week 26 it was further reduced to 85 kHz. In week 32 it was relaxed again to 105 kHz. The grey bands correspond to Machine Development (MD) and Technical Stop (TS) periods.

- *Detector conditions:* an application is needed to access and update/publish detector conditions
- *Commissioning:* it should be possible to run alignment and calibration tasks offline during the commissioning period, with the possibility to update the detector constants in the condition database.

The alignment and calibration tasks require only a small part of the CPU power of the EFF and can run concurrently with the HLT applications without affecting their throughput.

2.1.4 Online streams

During Run 2 data taking, the data selected by the trigger are organized in two main streams:

- a `Full` stream, in which the RAW events are reconstructed offline, and subsequentely filtered, streamed and possibly reduced in size (stripping), according to selection criteria specific for each physics analysis (stripping lines).
- the `Turbo` stream, described above. This stream is turned offline into a μDST format and made available to analysts.

Other smaller streams are retained for specific physics studies (no bias samples, low multiplicity events for Central Event Production studies) as well as for monitoring and calibration purposes.

In the `Turbo` stream, the output of the reconstruction is packed, serialised and persisted in a raw bank. Each HLT2 selection in the `Turbo` stream can choose the level of information to be

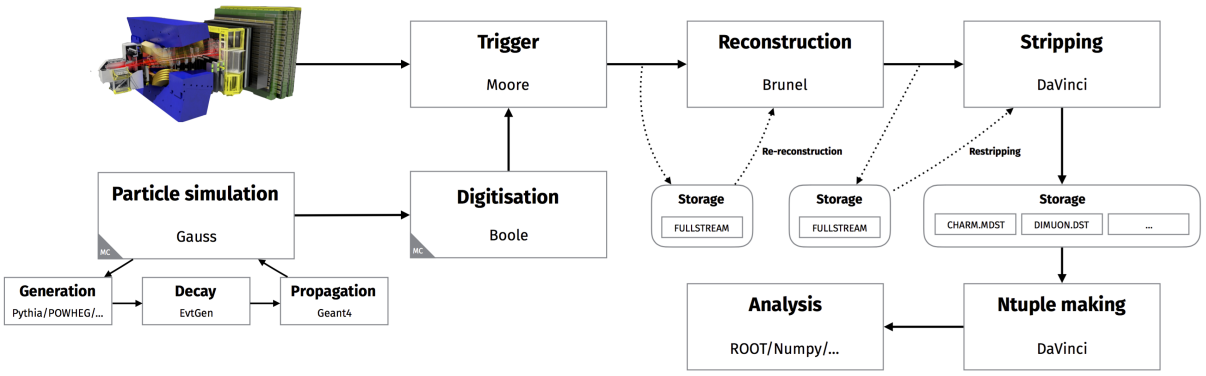


Figure 2.3: The LHCb data flow and data processing applications

recorded, ranging from particle candidates of the signal process under study (composite objects or basic particles such as a pion or a muon) to additional information such as the ouput of algorithms that analyse the rest of the event, to the entire HLT2 reconstruction. This concept of fine-grained, per-selection controlled selective persistence enables the optimization of the trigger bandwidth, and consequently of the offline storage resources, without sacrificing contents needed for physics analyses. It still requires verification before deployment, such that no important information in the event is discarded. The conversion from information from the persisted objects to analysis objects such as tracks, calorimeter clusters and particles, is performed by the Tesla application as detailed in the next section.

2.2 Offline data processing model in Run 2

The LHCb data processing applications for Run 2 are shown in Figure 2.3. Simulated events are handled by two applications, Gauss and Boole. The former performs the event generation and tracking of particles in the LHCb detector, the latter simulates the detector resonse. The Moore (*i.e.* HLT) application is then run offline in order to emulate the trigger response, after which the same path is followed for both real and simulated data; the event reconstruction is performed by Brunel, while DaVinci performs higher level tasks such as filtering, ntuple making and physics analysis.

2.2.1 Stripping, streaming and offline selections

After the offline reconstruction of the RAW events of the FULL stream with Brunel, datasets are *stripped* according to selection criteria organised in *stripping lines*. The output of several (O(100)) stripping lines is then merged in about 10 streams according to their physics contents. Streaming is needed in order to keep the amount of data to be processed by each physics analysis at a manageable size. The merging of several stripping lines in a small number of streams optimizes data management, though keeping a reasonably small overlap between streams. The output of the streaming process can use two formats based on Root [13]:

- DST, where the full event is saved, or

- μDST, where mostly information concerning signal candidates is saved.

The online format of the `Turbo` stream is stored in a format (*mdf*) that is optimized for space and speed (pure bytes stream and not ROOT-format). These data are then converted to a ROOT-based format by the TESLA application. TESLA decodes the raw data, converts them to event model objects, splits the data into several streams and persists the information as packed containers that can then be used in user analysis jobs. TESLA also performs the handling of events used to compute the luminosity corresponding to the data file and inserts the luminosity information in the file summary record.

When running on simulated data, TESLA also creates relation tables to match charged particle and calorimeter cluster objects to Monte Carlo particle objects. For μDST output, where only signal particles are persisted, the list of Monte Carlo particles and Monte Carlo vertex objects is filtered based on the simulated signal process and the Monte Carlo objects that can be associated to the HLT2 reconstruction. Streaming in TESLA is implemented using the same framework used for μDST streaming, whereby HLT2 lines are grouped into streams and each stream generates an output file that contains only the information recorded by the HLT2 lines belonging to that stream.

2.3 Online data processing model in Run 3

For Run 3, the data processing model is logically unchanged with respect to Run 2, except that it is foreseen to move almost all physics selections to the `Turbo` stream, which means that stripping selections will be to a very large extent moved to HLT2. The main change in the Run 3 data taking is the suppression of the hardware level trigger (L0) and the implementation of a full detector readout at 40 MHz. The HLT reconstructs events at the corresponding 30 MHz inelastic proton-proton collision rate and performs the appropriate selections.

As in Run 2, the HLT consists of two stages, HLT1 and HLT2. The former performs a fast reconstruction of tracks and applies selection criteria based on global event cuts as well as on the transverse momentum and impact parameter of tracks. The HLT1 must stand the collision rate of 30 MHz and perform an efficient rate reduction down to about 1 MHz. As in Run 2, it writes the selected events to a disk buffer. Calibration and alignment is then performed so that the resulting parameters are immediately available and used by the HLT2 that runs asynchronously. This second stage aims at achieving an offline-quality event reconstruction and applies selection criteria, implemented in trigger lines, that are tailored to each individual physics analysis.

Since the input rate of HLT2 is a factor about 30 smaller than that of HLT1, more elaborate and time-consuming algorithms are allowed in HLT2. In contrast, the HLT1 runs up-front, and this is therefore the most time-critical application that needs to be optimised for speed.

2.3.1 `Turbo` and online streaming

As discussed in Sec. 2.1. the `Turbo` stream removes unnecessary information early in the data flow, with a clear benefit for offline storage resources. The `Turbo` stream will become the default in Run 3 and will be used in the vast majority of selections, except for few specific use cases (Sec. 2.4.1), where the full event information and offline processing are still required. The `Turbo` data at the output of HLT2 will be arranged in a single stream. The output format of each trigger line will be customised, as in Run 2, according to the required level of detail, and a

bandwidth division mechanism similar to that used in Run 2 will optimize the rate for each line and the efficiency for physics, while keeping the throughput below its maximum allowed value.

2.3.2 Data-quality monitoring

In the Run 2 data taking, given that part of the analysis is performed in real-time on `Turbo` data, part of the data quality validations that were traditionally performed offline were moved online. Each of the automated calibration and alignment tasks was equipped with a dedicated monitoring algorithm which tracked the evolution of the constants produced by the task in question. Each of these constants had an associated threshold, which represented the maximum allowed variation from one iteration to the next, and which was set based on detailed studies of how these constants had varied during Run 1. If a constant changed by more than this threshold from one variation to the next, it would be flagged as requiring an update, while if a constant changed by too much from one iteration to the next, an expert was called to verify the output of the alignment or calibration task. For Run 3 it will be necessary to further improve online monitoring of alignment and calibration by transferring at least some of the offline data quality checks online. It will also be necessary to use the knowledge accumulated during Run 2 to maximally automate these data quality checks, in order to maintain a manageable workload for both shifters in the control room during data taking and software experts. Commissioning of monitoring and quality assurance will require a dedicated study using misaligned simulated events.

2.4 Offline data processing model in Run 3

As in Run 2, the same reconstruction will be run in the online and the offline environments. The LHCb software framework Gaudi enables to arrange the same algorithms to build either algorithm sequences in the online application Moore or in the offline reconstruction application Brunel (that must also be executable on simulated data).

Unlike in Run 2 where the online HLT selections and offline stripping frameworks are distinct, in Run 3 a single selection framework is envisaged, taking as a starting point the existing HLT selection framework. For the offline use it will be necessary to package this in an application independent of Moore that also supports offline specific use cases, such as:

- ability to add Monte Carlo truth[3] information;
- ability to run in flagging mode rather than rejection mode, for trigger emulation purposes in simulation; note that this is required also for the HLT selections;
- ability to refine the selective persistence to write out only the parts of the input data that are necessary for the analysis of the selected events.

The Run 2 stripping application is also performing streaming of the selected events. In Run 3 this will be performed by Tesla as in Run 2, since the `Turbo` stream will be the default. Streaming in Tesla and the selection framework should be packaged in such a way that they can be invoked from a single application if necessary.

[3]Monte Carlo truth is the fully known information on particles at generator and simulation level, in contrast to the *reconstructed* information available in data

2.4.1 Other use cases for offline reconstruction

There are other potential use cases that may require an offline reconstruction to be executed on full RAW data. For example:

- redoing the reconstruction offline to compare the result with the online version, for alignment and calibration studies;
- instrumenting the reconstruction for detailed monitoring of intermediate results (*e.g.* for online monitoring or reconstruction studies);
- using MonteCarlo truth information of simulated events for reconstruction or efficiency studies;
- offline reconstruction of streams that can not be reconstructed online (as made in Run 2 for *e.g.* heavy ions collisions data);
- offline re-reconstruction of streams that require a more complete reconstruction than what is possible in the online timing budget.

A dedicated offline reconstruction application (Brunel) will be retained for such use cases, to avoid unnecessary complexity in the mission-critical online application (Moore). The requirement that the configuration of the reconstruction sequences be packaged so that they can be run identically in the two applications (see above) avoids divergence of the online and offline reconstruction.

2.4.2 Offline streaming

By keeping the same granularity as in Run 2, based on runs corresponding to about one hour of data taking and several files of 15GB each for the `Turbo` stream, the Tesla application will first produce O(10) streams, each of which aggregates several HLT2 lines with similar physics contents. These streams will be further processed offline, by merging about 10 input files per stream and by further streaming by another factor 10. This will result for each run in O(100) streams, each of which will be made of files averaging again 15GB in size. These can be further merged together to reduce overheads. For operational reasons, streams should be as even as possible in size and the overlap between streams would remain at a reasonable level ($< 20\%$).

2.4.3 Working Group productions

The offline streams are input to the subsequent data selection and analysis steps that lead to the final fits and measurements. The DaVinci application will be used for that. The requirement in this area is that users should get the results of this data selection quickly. While Root ntuples are at present the preferred format for the final analysis steps, and it is foreseen that they will still be heavily used in Run 3, the possibility of using more compact formats as μDST or *mdf* files will be explored for resource optimization. The possibility of exporting data in a python-friendly format not based on Root will also be explored.

Currently, ntuples are produced by users in an unscheduled activity by running over `Turbo` or stripping output. Nevertheless, centrally-managed working group productions have been successfully implemented in the Run 2 data taking by exploiting the robust tools developed

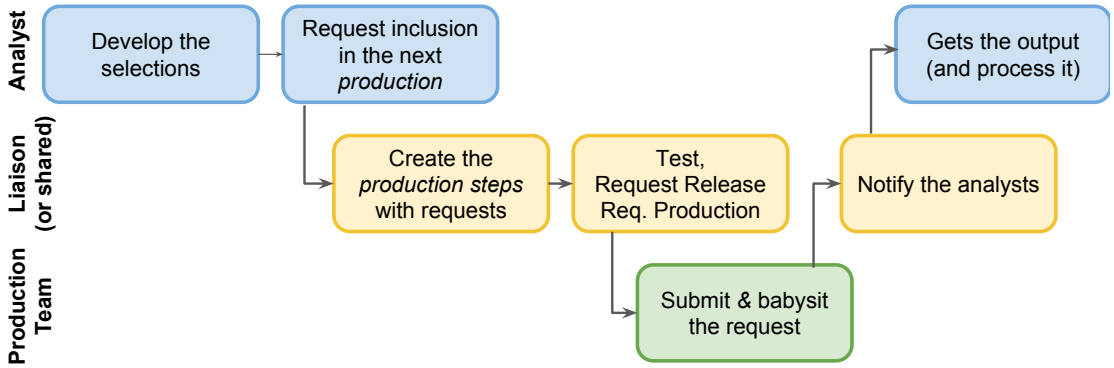


Figure 2.4: Schematic view of working group productions

in the distributed computing domain. They can be configured to reduce the analysis datasets to be tailored on the needs of a single analysis. They scale much better with the size of the input dataset with respect to traditional user jobs used so far. A schematic view of the current implementation of a working group production is shown in Fig. 2.4. The analyst develops and tests the selection criteria and releases the corresponding code, that is subsequently included in the following production cycle. The production team then takes care of creating the production steps in the LHCb distributed computing system (DIRAC, see Chapter 7) with the available requests, performs a final test and submits the production as in a normal production activity. When the production is finished, the analyst is notified and gets the output registered in the same catalogue that is used to store production data.

The output of the productions can be either in the ntuple or other formats. Using LHCb-specific format such as (μ)DST directly for physics analysis is feasible but currently unpractical, as this requires the entire LHCb software stack to be installed on a user laptop or desktop computer. The development of a light-weight library and extensive documentation would greatly improve the situation.

Working group productions will be the default mechanism for data analysis. Nevertheless, the possibility for single users to submit limited productions for analysis prototyping and testing purposes will be preserved by retaining tools such as GANGA [14].

The possibility to have a scheduled, centralised production organised in *trains* that "depart" on a given schedule, with analyses split in *wagons* each of them corresponding to a working group production, has also been considered, based on the work done within ALICE [15]. While appealing from the point of view of data management and resource optimization, this approach presents an implementation overhead that needs to be evaluated in detail.

Chapter 3

Core software

The development of high-throughput software, needed by the trigger in order to cope with the 30 MHz input rate from the detector and provide a fast decision, is coupled to the evolution of computing hardware in at least two aspects:

- the specifics of hardware technology determines the extent and type of parallelism the software can leverage;
- the trend of computing hardware evolution must be accounted in designing *scalable* software architectures.

Even though the number of transistors on integrated circuit chips has continued to follow Moore's law over the last decade, and will probably for the next few years, the trend of ever faster single-threaded CPU performance has been broken about ten years ago. Instead, the additional transistors have been invested into multiple CPU cores on a die, and, within these cores, into wider execution units. Furthermore, the gap between the processing units (PUs) and memory has increased, and feeding the processing units with data is becoming more than ever a bottleneck.

To address the stringent requirements on the data processing throughput, the utilization of computing capacities of current and emerging hardware has to be improved both quantitatively and qualitatively. The latter must encompass functional improvements (engaging previously unused dimensions of computing resources, e.g., threads, co-processors), as well as non-functional ones (ensuring the vertical scalability of the framework and its applications).

The Gaudi framework [11] is the common infrastructure and environment for the software applications of the LHCb experiment. It was designed [16, 17] and implemented before the start of the LHC and the LHCb experiment, and it has been in production without major modification ever since. Even though the main design principles remain still valid, a review and modernization of Gaudi is needed to meet the challenges posed by the Run 3 upgrade of LHCb and to give it the necessary flexibility to adapt to forthcoming challenges.

This chapter is organised as follows. Sec. 3.1 reports the changes that were necessary in the Gaudi software framework in order to better exploit the modern computing architectures in view of the Run 3 processing. A few examples of exploring other directions, such as code modernization, vectorization and the optimization of data structures are described in Sec. 3.2. The restructuring of the event data model, described in Sec. 3.3, represents a crucial step in order to optimize the data structures for algorithmic speed, while facilitating data analysis at

the same time. The evolution of two other crucial core software components, conditions data and detector geometry, is discussed in Sec. 3.4 and Sec. 3.5, respectively. Finally, a summary of the requirements for core software in the online environment is given in Sec. 3.6.

A benchmark of the trigger software application is reported in Chap. 4, clearly indicating that significant speedups can be achieved by following these different directions. The possibility of using non-x86 architectures and co-processors in the EFF is discussed in Chap. 5.

3.1 The GAUDI Framework and its evolution for Run 3

The GAUDI framework was designed following a data processing model where events are processed sequentially. Both the algorithm sequences, as well as GAUDI algorithms within each sequence, are executed sequentially for each event. With such a data processing model, only a multi-process approach is possible, in which all processes (or *jobs*) are executing the same application. Process forking may be used in order to reduce the memory footprint of the set of processes, and this is the approach that has been used so far in the EFF for Run 1 and Run 2. This approach suffers from several fundamental limitations, that affect the throughput and scalability characteristics of the framework on modern computing architectures. The major limitations are:

- weak scalability in RAM usage;
- inefficient handling of CPU-blocking operations, like *e.g.* disk/network I/O or offloading of computations to co-processors, e.g. GPGPUs.

To mitigate the above-mentioned limitations, the following techniques and models have been considered in the context of GAUDI:

- simultaneous multithreading;
- task-based programming model [18] (by which the basic computation units, *e.g.* the GAUDI Algorithms, are represented and handled as tasks),
- concurrent data processing model comprising inter-event concurrency (simultaneous processing of multiple events), intra-event concurrency (simultaneous execution of multiple GAUDI algorithms within each event being processed), and (optional) intra-task concurrency (decomposition of a task, like for example a GAUDI algorithm, into smaller sub-tasks).

The sustainability and conformity of these principles were demonstrated by a prototype [19], developed as an extension of the current GAUDI framework. The new paradigm required the development of new GAUDI components. Among them, the GAUDI task scheduler [20] singles out by both functionality and complexity and is central to the operability and viability of this new approach [21].

In the multi-threaded approach the programming paradigm is changed. One single instance of an application is run on a microprocessor and the program handles the parallelization on the cores. The guiding design principles in such approach are:

- thread safety and re-entrance, that are compulsory to ensure the general functional correctness of the framework and of its applications in a concurrent environment, as well as to achieve good scalability;

- the declaration of data dependencies, that are needed by the task scheduler for task concurrency control;
- the immutability of data, that significantly simplifies the concurrency control by allowing to focus on resolution of control and data flow rules.

3.1.1 Implementation

The implementation of a concurrent, task-based framework according to the above design principles, as well as the upgrade of all associated data processing applications, requires a deep revision of the corresponding LHCb code base. A re-factoring of many existing components was made to make them compliant with the requirements of thread safety and re-entrance. The mechanisms of declaration of data dependencies between the Gaudi Algorithms call for other wide-ranging changes. In particular, their input and output data requirements must be explicitly declared to the framework. Algorithms have thus been categorized according to this, and common interfaces and implementation rules for each category have been provided, thus allowing code developers to integrate their algorithms in the task-based framework with minimal efforts. In order to be able to use multi-threaded applications in Gaudi, a wide campaign has been launched to systematically identify the non-thread-safe code in the LHCb software stack and fixing it. The implementation of the design principles described above in the new software framework greatly simplified both the coding of algorithms and the checking of the thread safety of the code.

The concurrent paradigm places several stringent constraints on the event data model, the most important of which is that data must remain immutable after it is made available. This implementation strategy of a new event data model is described in Sec. 3.3.

3.1.2 Memory footprint

The multi-process and multi-thread approaches often lead to similar results when the parallelism is low (up to 10 threads) or when the memory footprint of the program is modest. However, the multi-process approach often suffers from the fact that the different processes are not sharing most of their memory, while the different threads of the multi-threading approach do. Memory can be an issue either by not being sufficient (thus inducing swapping) or because of the inefficient use of CPU caches.

The memory size issue is particularly worrisome, Fig. 3.1 shows the evolution of processors over 40 years and in particular the increase of the number of cores per processor in the last 10 years. Historically, the memory market has been the most volatile of the major IC product segments. Fig. 3.2 shows that the average selling price (ASP) for DRAM memory has more than doubled in just one year. If this trend is confirmed, it is clear that memory per core will become a scarce resource that should be optimized in order to reduce computing costs.[1]

The event reconstruction of the HLT1 application (see Chap. 4) has already been migrated to the new framework and has thus been run in a multi-threaded environment. Figure 3.3 shows the comparison of memory consumption between the multi-process and multi-threaded case. Tests are run on a machine with 20 physical cores and 40 hardware threads (2 per core).

[1]Intel is currently manufacturing processors with up to 72 physical cores and 288 virtual cores. For a server featuring 128 GB of memory, this corresponds to only 400 MB per core.

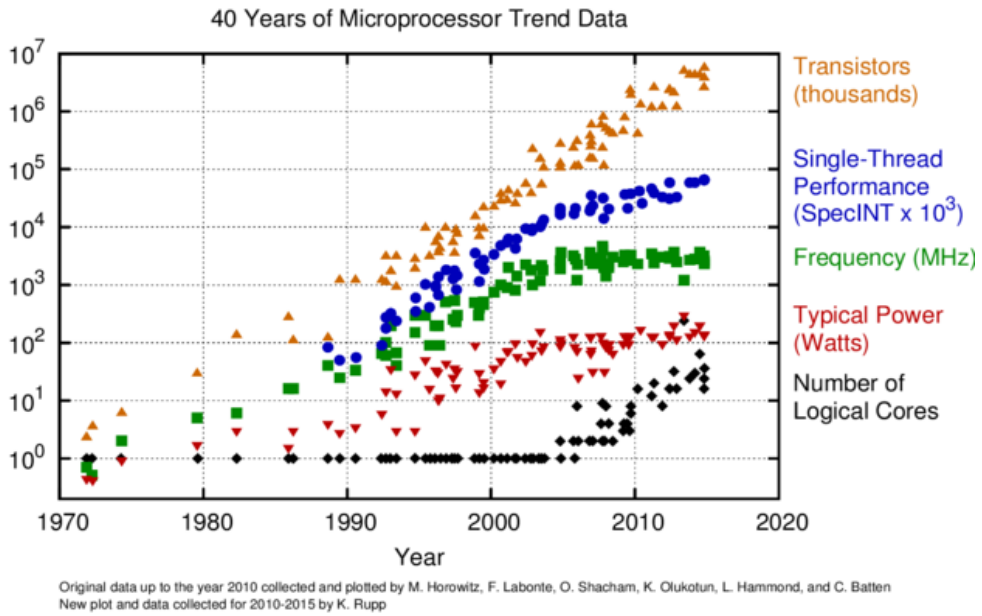


Figure 3.1: Evolution of Microprocessors over 40 years. Taken from [22]

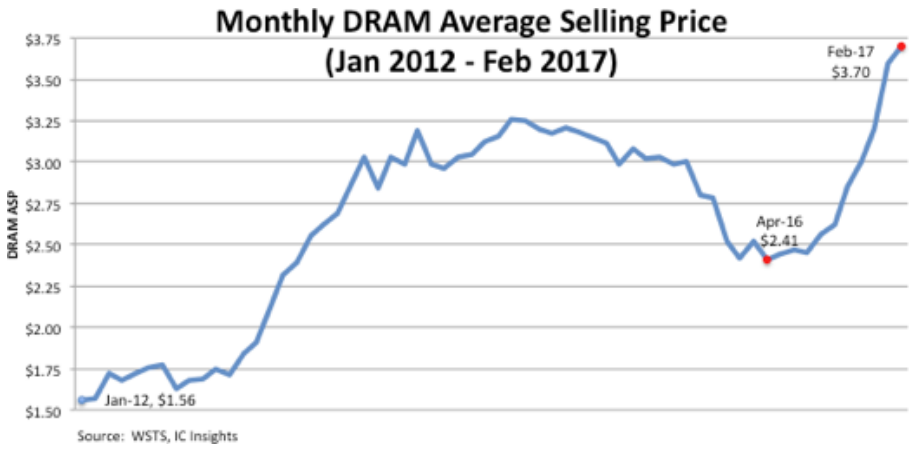


Figure 3.2: Evolution of average selling price (ASP) in US dollars of DRAM in five years (from January 2012 to February 2017. Taken from [23]. The ASP further increased by another 40% from February to July 2017 [24].

In the region where the number of processes or threads is high (and where the throughput is maximum, see Chap. 4), the memory usage in the multi-thread approach is a factor of about 40 smaller than the memory usage in the multi-process approach.

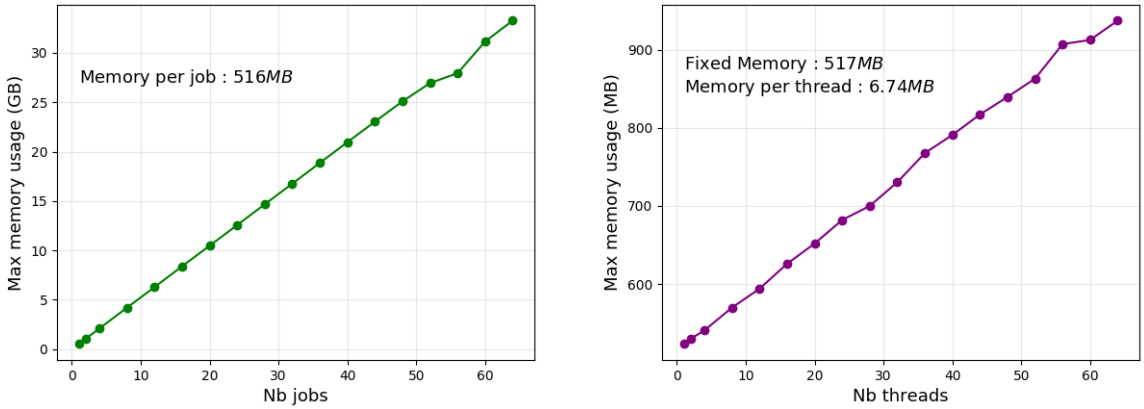


Figure 3.3: Memory consumption for multi-job and multi-thread approaches: test run with 3000 events per thread on a machine with 20 physical cores.

3.2 Other directions

In addition to the introduction of multi-threading at the core software level and to the improvements of the underlying core framework, other directions have been actively pursued in order to optimize the performance of the LHCb software:

- code modernization, using latest C++, including a full code review and the suppression of code that is no longer used;
- introduction and generalization of vectorized instructions in the computationally intensive parts of the code;
- review of the data structures, to make them more suitable for current memory architectures, in particular for effective cache usage and vectorization;
- potential algorithmic improvements in the most time-consuming algorithms;
- the usage of NUMA techniques.

General prescriptions and examples are given in the following for each of these items.

3.2.1 Code modernization

The LHCb framework is essentially written in the C++ language, namely with the version based on the official specification made in 1998 (C++98). During the following 13 years, the C++ language was left untouched, until a major step was done in 2011 with the adoption of the C++11 standard that introduced a lot of improvements targeting mostly efficiency and ease of coding. In the meantime, C++14 and C++17 have come and bring even more new features.

The LHCb code base had not been using these extensions of C++ so far, partly for backward-compatibility reasons, partly due to lack of efforts and knowledge. The upgrade of LHCb offers the opportunity to improve the LHCb code base and benefit from new performance improvements, e.g. by using extensively the move semantic paradigm.

This implies a complete code review of millions of lines of code and will thus need a major effort. However, this is also the occasion to clean up some parts of the code that have become

obsolete and can be dropped or simplified. This process is also encompassing the core framework itself, GAUDI, that is maintained and developed jointly with the Atlas experiment. An intense program of training (Sec. 8.3) has started and will continue in order to expose the LHCb collaborators at large to this paradigm shift in programming language and techniques.

3.2.2 Vectorization

Another major evolution of the hardware in the past years is the introduction of vector units in microprocessors. In practice, common mathematical instructions can be executed on vectors of integers, floats or doubles rather than on unique items within the same time frame. The currently most used vectorization units include SSE4, AVX2 and AVX512 with respective vector widths of 128, 256 and 512 bits. This means that an AVX512-enabled processor can compute 16 floating points operations in parallel on floats, as a float is 32 bits, or 8 double operations in parallel as a double is 64 bits.

Table 3.1: Potential ideal gains of vectorization

	SSE4	AVX2	AVX512
floats / int	4	8	16
doubles	2	4	8

This technology can obviously bring important speedups for the computing-intensive parts of the code (up to factor 16 in an ideal case on AVX512, see Tab. 3.1 for details), but has a cost in terms of software: the data organization has to be adapted to be suitable to vectorization. For example, the numbers in a vector have to be collocated in memory which is usually not the case by default.

The reconstruction code of the RICH subsystem was one of the first to be fully converted to the new LHCb framework, with particular emphasis to vectorization. All RICH algorithms are thread-safe and the most computing-intensive ones, in particular the photon reconstruction, have been vectorized by using an external library called Vc [25]. This allowed to take full benefit of the vectorization units on SSE- and AVX-enabled machines as shown in Tab. 3.2, where a maximum speed-up of a factor 7.9 on the computing part of the algorithm has been achieved.

One of the potential issues of this kind of approach is that the gain in the algorithmic part is often associated with a loss in the memory-handling part (allocation, deallocation and shuffling of data). The final speedup is thus a balance and cannot reach the ideal values that Tab. 3.2 depicts.

Another domain to be explored for substantial performance improvements is the possibility to run algorithms with single rather than double precision. As showed on Fig. 3.1, this gives a potential factor 2 gain on vectorized code, providing that the physics performance stays the same. This has to be validated algorithm by algorithm.

3.2.3 Data model

An extensive campaign of timing measurements of the LHCb software stack was performed. One of the main lessons learnt has been that the current LHCb data model is not suitable to the

Table 3.2: Performance of the vectorized RICH ray tracing

		SSE4		AVX2	
		time (s)	Speedup	time (s)	Speedup
double	scalar	233.462		228.752	
	vectorized	122.259	1.90	58.243	3.93
float	scalar	214.451		209.756	
	vectorized	55.707	3.85	26.539	7.90

memory architectures of current processors. Memory allocation has become expensive compared to computing and is a potential contention point when many jobs/threads are running in parallel.

This is particularly emphasized in case of many small memory allocations like in case of vectors of pointers to small structures, which are widely used in the LHCb code. Replacing this kind of structure by a vector of structures and preallocating it can give important gains.

Table 3.3: Top CPU consumers in HLT1; test run with 16 threads on a machine with 20 physical cores.

Function	CPU
`operator new`	16.7%
`_int_free`	8.3%
`PrPixelTracking::bestHit`	5.7%
`PrForwardTool::collectAllXHits`	5.7%
`PrStoreFTHit::storeHits`	4.5%
`PVSeed3DTool::getSeeds`	2.7%

Table 3.3 gives an idea on the impact of the current data model and the subsequent memory allocations on the HLT efficiency. It shows that the top consumers of CPU time are the `new` and `free` operators, with the actual physics code coming next and taking much less CPU time. One of the main providers of small allocations and scattered memory in the LHCb HLT1 application was found to be the use of LHCb-specific containers of pointers to objects (`KeyedContainer`) instead of those of `STL`, which would offer proper reservation of memory.

In order to measure the improvements that a cleanup of the data structures along these lines can give, the use of LHCb-specific containers in the algorithm performing pattern recognition of charged tracks in the VELO subdetector (`PrPixel`) was replaced by `STL` containers and memory reservation was added. Table 3.4 shows the improved behavior compared to Tab. 3.3 in terms of amount of time spent in the memory allocation. The gains are important and are coming from two effects: less contention on memory allocation and less time spent there, but also better cache

Table 3.4: Top CPU consumers in HLT1 after the removal of `KeyedContainers`; test run with 16 threads on a machine with 20 physical cores

Function	CPU
`PrPixelTracking::bestHit`	19.2%
`PrForwardTool::collectAllXHits`	9.9%
`operator new`	8.9%
`PrStoreFTHit::storeHits`	7.9%
`PVSeed3DTool::getSeeds`	5.5%
`_int_free`	5.1%

efficiency.

In conclusion, code optimization must include the minimization of the amount of time spent in memory allocations. This can be achieved by using adapting the choice of containers used, by pre-allocating large memory blocks, and by using vectors of structures instead of vectors of pointers.

3.2.4 Algorithmic improvements

The different kinds of code improvements that have been reviewed so far are purely computing-related. It is implicitly assumed that the output of the LHCb physics program (and in particular the trigger performance) will not be affected by these changes. This assumption is of course limiting somehow the spectrum of improvements that are being considered, and limits in particular changes to the core physics algorithms. However, this is where the potential gains are the largest, although the implications for the physics performance need to be carefully reviewed. Even larger gains could potentially be obtained by completely changing the strategy of the most time consuming algorithms, such as the VELO tracking one. An example of the performance improvements that have been achieved along these lines in the HLT1 application is shown in Chap. 4.

3.2.5 Optimized CPU usage with NUMA techniques

Modern processors operate considerably faster than the main memory they use. Non-uniform Memory Access (NUMA) techniques attempt to address this problem by providing separate memory for each processor slot, avoiding the performance hit when cores of the same processor slot attempt to address memory bound to this processor slot, as shown in Figure 3.4. It is worth noting, that the NUMA technology is strongly coupled to the choice of the processor architecture and hence to the hardware chosen for the EFF farm.

Usually the data do not all end up confined locally to one processor slot. Moreover processes may be served by all cores in all processor slots of the host. To ensure locality of both memory and CPU cores, the NUMA technology provides software which allows to confine processes to a

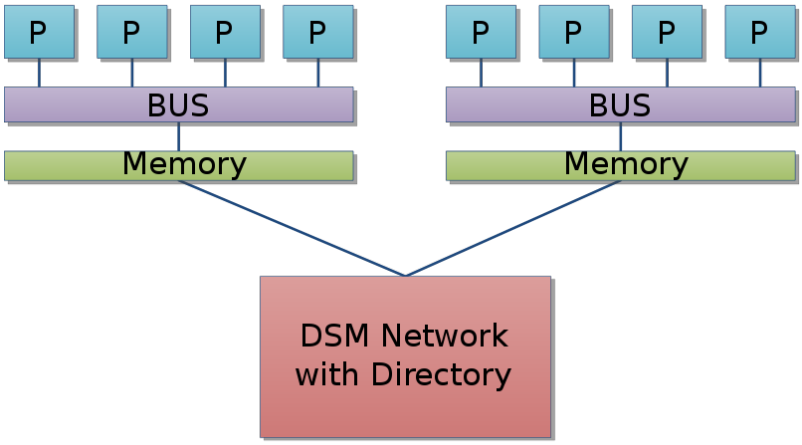


Figure 3.4: A graphical presentation of the NUMA architecture. The CPU usage is most efficient if any executing process is bound to one physical slot and only uses memory resources attached to this slot.

given slot.

To take advantage of this behavior, for applications where multiple instances of the same application are supposed to be executing with maximal performance such as HLT1 or HLT2, two approaches are possible:

- The slot binding is handled preemptively and one process is configured for each slot.
- One process is configured for all slots. This instance then replicates itself according to the number of CPU slots using a fork mechanism. Each of these instances then binds itself to one specific processor slot using the NUMA software API.

If process parallelism is desired these basic instances may further replicate themselves using a fork mechanism. The binding attributes hereby are inherited by the child processes. When threaded parallelism is envisaged, these slot-bound processes could start processing events.

Whereas the preemptive slot binding comes for free and only requires the NUMA command line interface, the optimized startup requires a Gaudi component binding the primary parent and child processes to the corresponding processor slots.

To properly incorporate the desired behaviour in a Gaudi process it is necessary to:

- Develop a Gaudi component encapsulating the required functionality to bind processes to a CPU slot.
- If only one single process per host node is supposed to be configured a forking mechanism must be provided. Such a mechanism should have framework support to ensure that at fork-time no open files are present in write/update mode, no other stateful connections are present and no threads besides the main execution thread is present.

3.3 Event model

The design and organization of LHCb event data is described here as *event model*. Ultimately, the event model should facilitate data analysis. Objects should enable the event reconstruction chain to function, trigger decisions to be made, and analyses to obtain the necessary information on reconstructed particles. In doing so, as little overhead as possible must be introduced when optimizing for speed as in the trigger applications.

The event model includes

- *transient data* that are only needed during the event processing to convey information between *logical* processing blocks, but are not stored in output files;
- *persistent data* that are stored in output files.

The event model does not include data that are "internal" within processing blocks, for example data that are used inside the RICH reconstruction and are exchanged between algorithms but that are not meant to be used outside the RICH reconstruction. Internal data are intentionally not regulated because the optimal data exchange format highly depends on the algorithms' implementations. In this sense the event model defines the data interface between processing blocks.

3.3.1 General requirements

Typically, the event data are organized in classes with relationships between those classes. However, given the computational aspects of Arrays of Structures (AoS) and Structures of Arrays (SoA), an explicit layout as classes is not enforced. Data layout as structure of arrays instead of a traditional array of objects can be beneficial for vectorization. Neither container format should be ruled out and the choice of the more appropriate should be made – upon benchmark results – on a case-by-case basis. As an example, the LHCb event model in Run 1 and Run 2 has almost exclusively used the `KeyedContainer` class as standard container format, and no SoA containers. This was shown to be inefficient for time-critical applications as the HLT (see Sec. 3.2.3). On the other hand, applications related to physics analysis require the usage of complex objects such as tracks and particles, that are inherently of the AoS type.

3.3.2 Transient data

Data are exchanged between algorithms using the Transient Event Store (TES) [11]. The data processing sequence can therefore be fully defined by the population of storage locations (algorithm output data) and the input requirements of algorithms. Individual algorithms do not require their data to be produced by a specific algorithm; they only require the storage location to be populated with well-formed data. The general requirements for transient data are listed in the following sub-sections.

3.3.2.1 Immutability

To enable multithreaded processing, data must not be manipulated once written to the TES. This becomes a serious constraint when dealing with references between objects (see Sec. 3.3.2.5). The memory address of an object may change before it is written to the TES, thus one can

reliably refer to it only after it has been written there. Bidirectional references are then a priori impossible: for example, when two objects are written to TES, the one that gets written as second can contain a reference to the first one, whose address is known. However, the object written first can not contain a valid reference to the second one, since that has not reached its final destination yet.

While foreseen for the Run 1 and Run 2 event model, immutability was not strictly enforced and not a technical requirement from the processing framework. Exceptions were allowed in the initial design, when copying data instead of manipulating single data members would have led to significant overhead.

3.3.2.2 Extendability

Event data should be *extendable* by additional data members: instead of either copying objects to update data members or manipulating data in the TES, additional data are supposed to be stored in separate data objects to extend the initial data. For instance, a track fit algorithm should publish the fit results on TES and algorithms using tracks should be able to read the tracks and the fit results as if fitted tracks would be read. It is therfore the task of the framework to ensure that reading of extended data is consistent: if there are two tracking algorithms producing tracks, and two corresponding track fit result data locations, it must not be up to consuming algorithms to combine the corresponding locations. This avoids a large source of potential bugs.

In the data layout as SoA, there is no run-time cost for combining a base data object and its extension to an extended data object. Data remains arranged in multiple arrays and combination happens at compile time when translating the user code.

In contrast to the Run 1 and Run 2 event model, *extendability* replaces the need to manipulate data in the TES. Furthermore, it allows to use light-weight objects in the HLT1, and to add data members during the processing to end up with "complete" reconstructed objects in later processing stages. Extendability is also solving one of the drawbacks of the current event model, in that simulated particles and reconstructed particles are separate classes, rather than simulated particles just being "normal" particles extended by MC truth information.

3.3.2.3 Iteration over multiple containers

It is necessary that different algorithms produce data of the same type and populate different TES locations. For instance, Monte Carlo generation will create Monte Carlo particles (`MCParticles`) in the generator phase (handled by *e.g.* Pythia, EvtGen) and in the detector simulation (handled by Geant4). It is not possible for these to be combined into a single TES location, as data are passed from the generation to the simulation step through the TES and thus populates two locations. As for subsequent algorithms it is not of any interest to know where particles were created, forcing users to correctly collect data from both TES locations introduces another source of potential bugs. Iterators of the `MCParticle` containers must thus be able to iterate over multiple TES locations. The Run 1 and Run 2 simulation addresses this need by manipulating the TES location of generator particles and adding the detector simulation particles to the container, however this is forbidden in the new model.

Both extendability and iteration over multiple locations are addressable through *virtual* TES locations, *i.e.* locations which contain nothing other than links to the TES locations which are to be combined for extension or for continued iteration.

3.3.2.4 Tools for managing transient data

The TES does not impose requirements on the data objects it contains. The most trivial implementation are thus `std::vector`s and can in principle be used throughout the data processing. As standard tools, they are highly optimized and debugged for performance and functionality, they support iteration over contents as well as random access, and can contain any custom C++ object. As mentioned before, such AoS storages may not be the most optimal when aiming for vectorization. The RICH reconstruction referred to in Sec. 3.2.2 profits from using the `Vc` and `range-v3` [26] libraries.

Similarly, the studies reported in Sec. 5.1.5 show that the vectorization of many algorithms requires data laid out in SoA structures.

A helper class `SoAContainer` [27] has been prepared to implement SoA data structures with interfaces like regular AoS structures, by means of template programming.

3.3.2.5 Referencing

Data containers of event model classes must support random access. In the case of AoS structures with `std::vector`s, this can be done as C++ pointer to the contained objects, as STL iterators, as any more powerful pointer class (shared pointer, unique pointer, ...), and as numeric index.

Such pointers are needed to extend information in objects beyond the extendability described above. Extendability makes sense when all input objects need to be extended, and the container of extensions has the same length as the container of extended objects. In the case where only a subset of an input container is extended, *e.g.* only a subset of all tracks is passed to a track fit, the extending objects must keep references to what object they refer to. Alternatively, the extending container can be forced to have the same length by filling "gaps" of tracks that are not fitted with zeroes or *NaN*s. In that case, care must be taken to not introduce any time penalties.

Given that such references always refer to immutable data in the TES, there is no need to deal with problems that arise when the location of the referred-to data changes (invalidating pointers) or that its container changes content (invalidating indices).

Pointers can be preferred to indices. Indices alone cannot be interpreted, require knowledge of the container (TES location) they refer to, and remain difficult to interpret when referred-to data reside in multiple containers.

In SoA structures, such pointers do not exist, as referred-to objects do not exist in memory and thus have no memory address. In the case of SoA storages, container indices must be used.

This means different container implementations can a priori not be replaced with each other transparently. This discourages frequent changes and emphasizes the importance of careful choices.

3.3.3 Persistent data

In the `Turbo` [5] paradigm, a key feature of the event data is that it is possible to store only part of the event: each trigger or filtering line has to specify which of the event data are relevant for further analysis, resulting in a subset of all tracks, clusters, particle identification (PID) and other physics objects to be persistified to output files. Writing multiple output files at the same time should also be allowed to separate data for different usages, such as real time calibration, offline reconstruction, monitoring, and streaming.

Consequently, each output file contains a different collection of persistified objects from the same original TES location. An object can thus be at different locations in a persistified collection, depending on which other objects are persistified. References to such an object must be updated during persistification. That means each persistification algorithm must keep a lookup of which original TES object ended up at which persistified location, and these lookup tables must be used when storing objects which refer to persistent objects. This requires persistification algorithms to be "aware" of which part of data objects are references.

As already mentioned in Sec. 2.1.4, selective persistification was implemented and commissioned during the Run 2 data taking, and it is already in use. For Run 3, it must be maintained and adapted to changes in the event model of transient data.

The separation of which data members of objects are *actual data* and which are *references* adds complexity to the design of persistent objects. To ensure this separation can be handled in persistification algorithms, the complexity of objects that are persistified must be restricted. Candidates to normalize persistified data structures are either PODIO [28] or the Gaudi Object Description [29]. The former is a C++ library based on the usage of plain-old-data (POD) data structures wherever possible, while avoiding deep-object hierarchies and virtual inheritance. This improves runtime performance and simplifies the implementation of persistency services. The latter, implemented in Gaudi and currently used in LHCb, is a system for event object description that derives the actual implementation of the event objects (in *e.g.* the C++ language) starting from their description in a high level language such as XML.

3.3.4 Data classes

In addition to implementation candidates for data container classes, and persistification algorithms, the "physical" data members of the event model classes need to defined and reviewed, to ensure all data that is needed for analyses (and also data quality, detector monitoring, detector alignment, simulation studies, and so on) are accessible. The classes in which the data are divided and their members can roughly follow the existing design.

The requirements for the charged particle track class have been collected in an internal document as a first step towards the implementation of the data classes and their relations. The subsequent design is based on a *one-size-does-not-fit-all* approach, where the data members are thinned down to what is needed in the reconstruction chain. Additional information that is required at a later stage (*e.g.* physics analysis), or might be helpful for monitoring or alignment tasks, are considered as extensions which will be added in the reconstruction only in dedicated applications. In this manner, the processing speed of time-critical application such as HLT1 will not be compromised by information that is only needed in a fraction of the data, *e.g.* that collected during the first minutes of collisions in a fill.

3.4 Conditions data

Conditions data represent information about the detector that is required for data processing (*e.g.* detector configuration, alignment or calibration constants, environmental observables). A given condition is valid for more than one event (typically many more, from minutes to years of data taking), but may not remain valid for the entire lifetime of the detector.

Conditions data have a fine granularity, down to the level of individual detector elements. The space of conditions data are effectively three-dimensional: a "spatial" dimension, the time

evolution, and a versioning applied to the entire conditions dataset. An overview of conditions data handling in LHCb is detailed in [11]. The remainder of this section focuses on the parts of the LHCb conditions handling infrastructure that will be changing in the context of the LHCb upgrade.

3.4.1 Time representation

As conditions data represent the time evolution of the detector state, they naturally have to build upon a digital representation of time. The historical timing representation of choice was based on real-time clocks with a nanosecond resolution [11]. However, the difficulty of keeping real-time clocks synchronised to that level of precision eventually led to the adoption of alternate counter-based time representations. The most popular time representation in LHCb is based on the counting of individual events and on the notion of a data-taking *run*, corresponding to a period of data taking with stable conditions, typically corresponding to a maximum of one hour.

The adoption of these new time representations however does not mean a complete deprecation of real-time clock timestamps: the LHCb conditions database infrastructure and the software framework are currently expected to mix and match both time representations, which is a source of unnecessary complexity. In the long term, it is desirable to eliminate this complexity by converging towards a single unified time representation across the entire LHCb conditions handling infrastructure.

3.4.2 Conditions database

Because conditions data can be valid for a large set of events, storing them inside of event data files would lead to a space-inefficient and error-prone duplication of information. Instead, conditions are usually centralized in a dedicated database, which is kept separate from event data files. This database is called the *conditions database*.

Conditions data are usually written once and read many times. Write rates for any given detector element are relatively low, ranging from once every few minutes to once a year, but the database-wide write rate can be higher as conditions writes pertaining to different detector elements are not always synchronized. Read rates can be much higher, up to thousands of requests per second, which requires to use an appropriate caching infrastructure. Concerning data volumes, the combination of the LHCb detector description and the set of all recorded LHCb conditions from 2008 to 2017 amounts to 2GB of storage, and the conditions database is expected to grow over time at a rate of the order of magnitude of one gigabyte per year.

The LHCb experiment uses conditions at every stage of its event processing workflow, from the high-level trigger to some specific forms of physics analysis. During Run 2, as detailed in Chap. 2, the LHCb high-level trigger infrastructure was upgraded in order to allow for new alignment conditions to be determined during data taking, by introducing a new real-time alignment and calibration system that operates after HLT1 [4].

After many years when conditions had been stored in a set of SQLite files that were accessed by means of the COOL [30] library, LHCb is using a Git repository for conditions since 2017. Such a solution:

- maps very well the three-dimensional space of conditions data, where the folder hierarchy represents detector elements, time evolution is represented using an index mapping a set of payload files, and dataset versioning is performed using version control commits and tags;

- offers a long, sought-for replacement for COOLL, that is unmaintained and with no plans for further developments, is not thread-safe, and presents scaling issues related to the increase in the number of clients.

No major modifications are foreseen for using Git-based repositories for conditions data in Run 3.

3.4.3 Conditions data payloads

The conditions data payloads that are fetched from the database are currently encoded in a dialect of the XML data format. This design comes with the disadvantages of XML, such as a high level of verbosity which slows down computer parsing and makes human inspection tedious, without leveraging the benefits of this data format, such as schema standardization and easy analysis using off-the-shelf tools. There is thus a longstanding requirement to investigate alternate conditions data payload formats.

In the context of the LHCb upgrade, one particular motivation for pushing this investigation further is the merging of the online and offline data processing frameworks. The long initialization time of the offline LHCb framework are highly problematic in an online context, see Sec. 3.6. One particular source of overhead which has been known for a long time, but has not yet been resolved, is the large amount of time spent processing conditions data during the startup of the framework.

The fact that extensive optimization work on the conditions database back-end did not resolve this issue suggests that other parts of the conditions processing pipeline could be responsible for this high processing time. One particular area which has been identified as requiring further investigation is the conditions payload format and the code used to parse it.

Resolving this long-standing performance problem would simplify the LHCb upgrade software framework and reduce the divergence between the online and offline data processing code paths by eliminating the need for additional software complexity such as caching and reuse of the fully initialized detector description.

In conclusion, a change in the format of the payload of conditions data is mandatory for both performance and maintenance issues and efforts should be devoted to that.

3.4.4 Framework interface

Conditions handling requires a significant amount of support code on the experiment framework side, performing tasks such as sending requests to the conditions database, parsing the conditions data payloads, storing the decoded conditions data in RAM, computing "derived conditions" such as detector alignment constants, making the result available to data processing code, and using caching techniques to avoid duplication of effort.

The LHCb-specific code performing these tasks was built for an era when the Gaudi framework only processed one event at a time, and heavily leverages this design assumption by modeling detector conditions as a set of global variables. This approach breaks down as soon as a single instance of the Gaudi framework is allowed to process multiple events at the same time, some of which are potentially associated with different detector conditions.

This issue has been mitigated by restricting the Gaudi framework's ability to process events concurrently through locking. That approach works well for the current conditions usage patterns of LHCb, where detector conditions change very rarely and are not used for offline data processing. However, as conditions are by nature used to handle unforeseen detector imperfections, it is not

known whether this current usage pattern will remain forever. In the longer term, a safer approach would be to rework the conditions handling code so that it is able to correctly manipulate multiple detector states at once.

This path is being explored in the context of an R&D project which aims to introduce a multi-threading-friendly conditions handling infrastructure in the GAUDI framework. This would simultaneously solve this issue and increase code sharing between the GAUDI-based experiments in the area of conditions handling, from core conditions handling logic to the integration in GAUDI of conditions-related components such as those of Ref. [31] or the conditions handling system of DD4HEP [32].

3.5 Detector description

The description of the geometry of the LHCb detector serves a wide range of applications, from simulation to detector alignment, visualization, computation of material budget. It must fulfill the LHCb needs in terms of flexibility, management, integration with the conditions database, and speed of navigation between detector elements.

The LHCb detector is described using an XML-based geometry framework [33], designed and implemented within the collaboration. This framework has also been used to describe the upgrade detector and is the base for the upgrade simulation. This code has served its purpose and has advantages: as it was developed in-house, it is completely under the control of the experiment. The downside is that development has stopped and no more improvements are implemented: LHCb is thus using a framework that is not optimized for modern architectures with SIMD capabilities.

Time-critical applications such as the HLT use for charged particle tracking a simplified version of the LHCb geometry, thus avoiding the overhead due to a full geometry description. However, the navigation speed of particles through the LHCb geometry could be improved in order to optimize the computing performance in simulation. The simulation application uses GEANT4 [7, 8], which uses a different geometry to propagate the particles through the detector. Some features are missing in the LHCb framework, that would be useful, *e.g.* replacing detailed volume descriptions by homogeneous ones where an average of the contained material is used, in order to speed up simulations.

While improvements could be added to the LHCb geometry, new possibilities have emerged since the framework was designed: it is now possible to use common geometry frameworks with the features needed by high energy physics experiment, e.g. the DD4HEP [32] toolkit.

3.5.1 LHCb and DD4HEP

Detector Description for HEP (or DD4HEP) is a toolkit that allows the full description of the geometry and comes with libraries for conditions management and visualization. The in-memory description is based on the ROOT [13] framework, but vectorized geometry frameworks such as VecGeom [34] can also be used.

Given these interesting features, the possibility to move to DD4HEP has been investigated. Feasibility studies have been performed by using a prototype converter of the LHCb XML geometry description into DD4HEP, provided by the DD4HEP developers. Some lessons have been learned:

- the expression evaluation in DD4HEP and in the LHCb code are done in different manners, leading to problems when loading the XML "as-is" using the DD4HEP DDDB plugin;
- the provided converter can be used for testing purposes, but it is not a final production solution: for this purpose, a redesign of the LHCb geometry layer is necessary.

Furthermore, a significant part of the upgrade detector was loaded in DD4HEP and the LHCb and DD4HEP in-memory representations were compared to ensure equivalence by either comparing the volumes and their position and size, or by checking the material volumes crossed on various segments). It was therefore possible to check:

- the integration of DD4HEP with the LHCb conditions database;
- the alignment functionality offered by DD4HEP and whether it is suitable to the operation of the LHCb detector;
- how to integrate the LHCb detector element classes with DD4HEP;
- how to implement the LHCb transport service with the ROOT geometry.

3.5.2 Program of work

The ongoing studies show that DD4HEP has equivalent functionality to what is provided by LHCb. Current efforts focus on the detector alignment functionality, the goal being to implement the LHCb alignment tests within the LHCbDD4HEP prototype by mid-2018. This will however leave some open issues regarding:

- the final persistency format for the geometry,
- the detailed validation of the full LHCb geometry, and
- the work involved in migrating the detector element code for each of the sub-detectors.

Indeed, the current converter is not suitable for production, due to the complexity of the XML structure and the time needed to load all the XML files. Direct conversion to the Compact XML format supported by DD4HEP is not easy or desirable either. It should however be possible to refactor the geometry to a new easily loadable format and write the appropriate module to load it into DD4HEP, as done by other HEP experiments such as CMS.

Once the migration of the LHCb geometry to a new format is done, the validation of the new representation is a time-consuming task that requires help and validation from all subdetectors. Several choices are however left open for the geometry description of the current LHCb detector, with the following constraints:

- In all cases, the representation of the current LHCb detector has to be kept accessible, as the LHCb trigger application used during data taking has to be used to process newly simulated data.
- Migrating and validating the geometry of the current LHCb detector represents a huge effort, similar in size to the migration of the upgrade geometry.
- Keeping the geometry of the current LHCb detector in the current format also means maintaining the associated software.

As the simulation software is most affected by these issues, the advantages and drawbacks of the various possibilities and the plan to reach a decision are discussed in Sec. 6.6.

3.6 Core software in the online environment

The seamless integration of the modernized GAUDI data processing framework in the LHCb online system and the steering of the data processing applications by the experiment control system requires a number of conditions to be satisfied. The additional functionality can typically be achieved by either extending an existing offline application and incorporating additional components, which e.g. support the steering of the application, or by replacing components also existing in the offline framework in order to e.g. adapt to the different mechanism to access event data or to redirect output messages to appropriate devices.

The following sections provide a condensed summary of these requirements. Past experience from the startup of the LHCb experiment also showed that adopting an existing framework for the use in the online environment at a later stage is more difficult and typically results in unnecessary compromises leading to a more error prone and less precise implementation in the end. For example, the simple fact that the initialization mechanism differs depending on the use of the application in the offline or in the online environment, caused numerous issues and frustrations when commissioning or debugging a process in the past. To avoid such unnecessary divergence between similar implementations it is indispensable to collect the requirements for the use of GAUDI in the online environment as early as possible.

3.6.1 Customizing GAUDI applications

The modernized GAUDI framework aims at configuring and customizing both the offline and online flavours of the application in a very similar manner and hence

- minimizing the differences between the offline and the online application,
- optimizing the re-use of code, and
- fully supporting the online functionality described in the following sections.

This requires that the mechanism to customize the individual components of the GAUDI data processing application for the online environment uses the same interface also used offline. Today this means to consequently support the application bootstrap using the generic startup script `gaudirun.py` together with the appropriate hooks provided by the GAUDI framework. Such hooks must include an interface which enables to synchronize the process configuration (see the following Sec. 3.6.2 for details) during the configuration of a run and must also allow to add/replace offline components with specific online components (see Sec. 3.6.4 for details). Using this mechanism any online derivative of a GAUDI application can be seamlessly configured. Though the topic is a shared issue between online and offline, its implementation is entirely covered by the core framework. Extensions should not be necessary.

3.6.2 Process configuration and scheduling in LHCb Online

All components participating in the data taking activity of the LHCb experiment are configured and initialized in a controlled and synchronous manner in several steps as shown in the

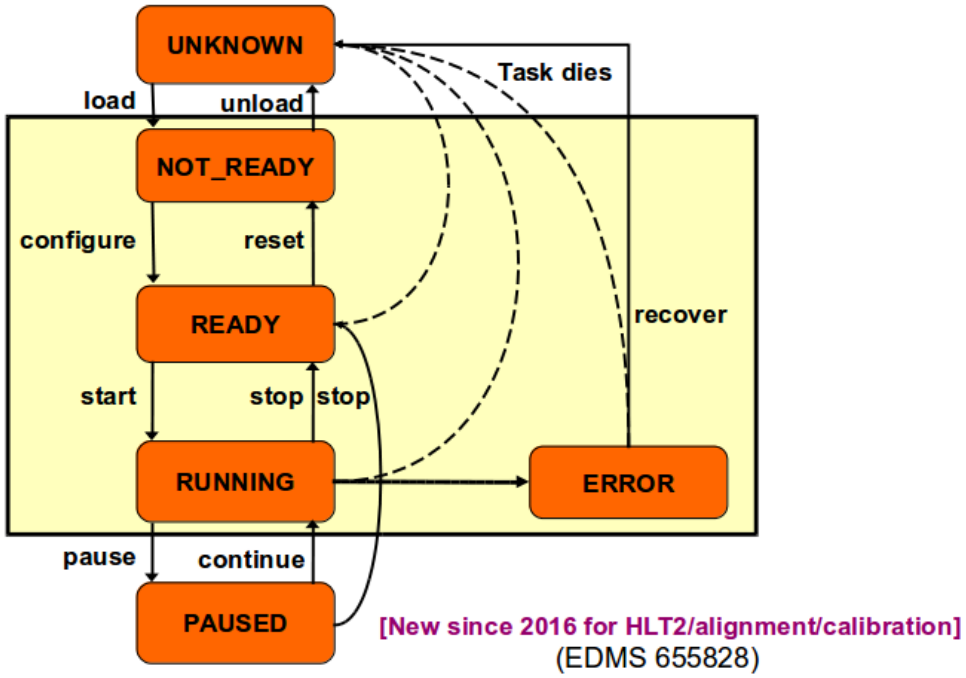


Figure 3.5: The state diagram and the corresponding transition between the states that all processes participating in the LHCb data taking activity must offer to the control system. Taken from Ref. [35]

Finite-State-Machine (FSM) diagram in Fig. 3.5, taken from Ref. [35]. This ensures that all participating components, independent of hardware or software origin collaborate well and do not e.g. prematurely expect the presence of services from other components.

The state diagram also roughly resembles the internal state diagram of GAUDI. Hence mapping between internal and online states should be straight-forward.

The execution of any path shown in the FSM diagram above is defined by the Run-Control program responsible for the overall synchronization of all components participating in the data taking activity.

The transitions between the different FSM states are triggered by external network stimuli. Clearly, a well defined master-slave relationship must be maintained at all times and, to (at least) approximate an ideal Petri net, all entities should be responsive at all times unless actively executing a transition, which requires finite time to complete.

Any data processing application implementing the FSM diagram in Fig. 3.5 must meet the following requirements:

- The startup path from UNKNOWN (i.e. the process does not exist) to the READY state and the shutdown path from the READY state to UNKNOWN state have to be executed exactly once.
- It is absolutely necessary that any other existing path between the READY, RUNNING and PAUSED state may be executed any number of times without compromising or corrupting the application. Applications executed during alignment or calibration activities strongly depend on the feasibility to freely navigate between these states.

It is necessary to realize that the time spent during initialization (i.e. the path from the UNKNOWN to the READY state) is firstly highly correlated to the overall configuration time of the LHCb experiment and secondly scales with the overall number of processes to be configured simultaneously. Any effort reducing this time has a significant effect on the overall data taking efficiency - in particular during a complete system re-initialization. Such re-configurations are a necessary consequence when the hardware configuration changes or during other recovery procedures.

It is envisaged to design and implement two components to meet this requirement: one for offline- and one for online usage, which navigate through the corresponding GAUDI states of the state diagram in Fig. 3.5.

Here the offline flavour would sequentially navigate the application from the offline / UNKNOWN state to the RUNNING state and subsequently after processing all input event data navigating back to the offline / UNKNOWN state.

The implementation used online will only invoke these transitions to the various states on request, depending on external network stimuli. An optional startup flag to automatically navigate to the RUNNING state shall be supported by the online component, which proved to be very useful in the past.

3.6.3 Dynamic load balancing

The new multi-threaded LHCb data processing framework must continue to support the dynamic load balancing on the EFF (Sec. 2.1.2) by *e.g.* dynamically increasing and decreasing the number of execution threads to ensure smooth running. Otherwise a change of resource usage would require to stop and re-initialize each process as in the course of a complete start-up. This would make fine-tuning difficult and very tedious. In addition a complete shutdown of an entire activity on the whole farm would considerably decrease the CPU efficiency.

3.6.4 Adapted behavior by replacing offline components

Some components must be replaced in exiting GAUDI-based data-processing applications to meet the consequences resulting from different processing environments between online and offline. The components to be replaced for the event data processing applications are:

- the `MessageSvc` implementing the `IMessageSvc` interface,
- the `MonitorSvc` implementing the `IMonitorSvc` interface,
- the `EventSelector` implementing the `IEvtSelector` interface,
- components handling processing errors during the event loop and
- components handling event data accepted by HLT1 or HLT2.

The following paragraphs describe the necessity to specialize these online components.

The reasoning to replace the `MessageSvc` with an instance optimized for online usage is simple: on each node where processes are executing in the context of a given activity (HLT1, HLT2,calibration, alignment, etc.) the output is re-directed to logging FIFOs [36], which accumulate all printout statements of the processes contributing to the activity for later presentation to the shift crew. Contrary to offline process execution, many processes contribute to the collected

output. Hence the replaced message service offers the possibility to conditionally suppress messages at a rather high level – a necessity resulting from the different sensitivities of developers contributing to online applications.

The `MonitorSvc` does not exist in the offline flavor of the framework. It is an indispensable component to extract counter-like information or histograms from the running application. This service is the main tool used to provide instantaneous monitoring information both at a low level to spot noisy and/or dead areas in the hardware of the detector as well as high level monitoring using reconstructed data. The existing functionality and infrastructure to support such a service and its accessibility by the algorithms must be preserved.

The `EventSelector` and the closely collaborating component, the raw data `ConversionSvc`, populate on demand the transient event store (TES) providing to all data processing algorithms which are TES clients a transparent mechanism to access event data, independent of the processing environment both offline and online.

Depending on the collaboration strategy between the online and offline replacement candidates either one or both of these components will have to be replaced with the online counterparts. The online `EventSelector` instance must be able to serve one additional functionality unique to its usage in the online environment: event access typically is blocking *i.e.* it succeeds once an event is available for processing – analogue to a blocking file-read. Stopping the data taking activity however requires to intercept such a blocking call. Such a functionality currently is achieved using the GAUDI incident mechanism from a separate thread with the consequence that after the return of the call to process the event, the reason of the failed call is lost. This mechanism should be intrinsically supported.

Processing errors may occur at any time during the process configuration, the handling of event data in algorithms or the access to supplementary information such as, for example, conditions data. In the online environment, where typically the processing of events must continue under all circumstances, it is very important in such an event to have access to information as precise as possible for taking the appropriate actions. These actions may not be identical to the appropriate actions taken when processing data offline: most of the time the error condition is linked to one single event and typically can be ignored. Under certain circumstances however the event loop needs to be stalled to avoid further loss of data and sometimes the process needs to be exited.

Events accepted by any of the HLT steps need to be declared to output buffers. This is currently done by a specialized instance of the algorithm class, which serializes the output banks in the appropriate form and declares it to the buffer manager [37]. Such an implementation should be straight-forward though being based on a different superclass. However, it can be assumed that major similarities with the current implementation exist.

Chapter 4

Benchmarking the HLT reconstruction

In the LHCb upgrade for Run 3, events from LHC collisions will be processed at a rate of 30 MHz by using a full software trigger. In order to meet this objective, described in detail in Ref. [2], the evolution of the HLT performance must be followed on a regular basis. The results of the most recent performance update [6] showed that HLT1, the first HLT stage that runs upfront synchronously with data taking, could process events at about 3.5 MHz only, due to a scaling of the performance of computing architectures per unit price that was worse than anticipated. Efforts were therefore prioritised to port the HLT1 to the modernized Gaudi framework (Sec. 3.1), in order to better exploit the previously unused dimensions of computing resources. The HLT1 algorithms that perform the event reconstruction were reviewed and converted in order to be run in a fully multi-threaded approach. Then, profiling tools were used to identify hotspots in the code and a first round of algorithmic improvements were implemented.

In this chapter, benchmarking tests of the reconstruction sequence of HLT1 are presented, with the purpose of showing that the efforts made so far significantly improve the throughput, thus demonstrating the validity of the technical solutions that have been adopted. Section 4.1 briefly illustrates the HLT1 sequence that was used for benchmarking, the results in terms of event processing rate and profiling of the algorithms are reported in Sec. 4.2, an outlook is given in Sec. 4.3. Further details can be found in Ref. [38].

4.1 The displaced-track reconstruction sequence

HLT1 processes events and reduces the event rate to a level that enables to run the full detector reconstruction in HLT2 while keeping the efficiency for physics signals as high as possible. The HLT1 reconstruction sequence considered in this study selects tracks or two-track vertices significantly displaced from the primary pp collision vertices, thus giving samples with long-lived charm- and beauty-hadron decays[1]. Fig. 4.1 gives a schematic view of this *displaced-track reconstruction* sequence. This reconstruction sequence includes the following logical steps (with the corresponding algorithm names in parenthesis):

- VELO tracks are found starting from VELO clusters (`PrPixelTrackingFast`);

[1] This corresponds to the *partial reconstruction scenario* described in Sec. 4.7.2 of Ref. [2].

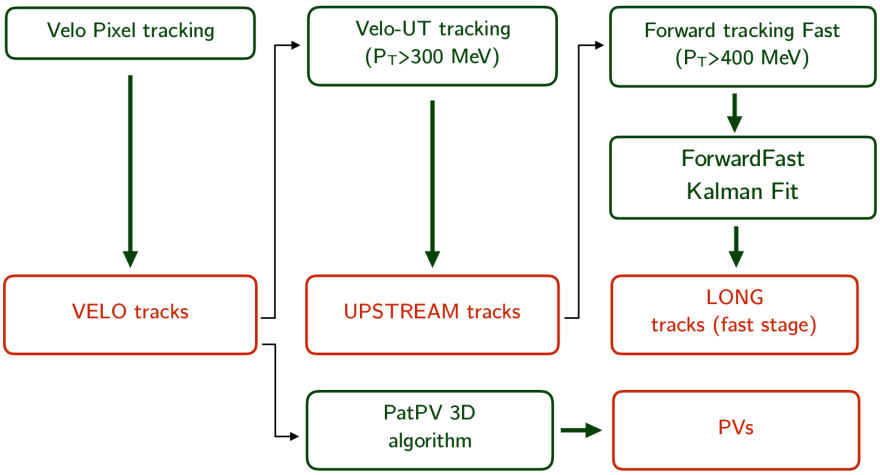


Figure 4.1: Schematic view of the tracking algorithms in the (*fast*) HLT1 stage. The transverse momentum thresholds are approximate and refer to the size of the search windows in the algorithms that perform pattern recognition.

- primary vertices are built from VELO tracks (`PatPV3D`);
- VELO-UT tracks are formed by extending displaced VELO tracks (with impact parameter greater than 100 μm from any PV) to the UT and searching for UT hits within a window corresponding to a possible track transverse momentum above 800 MeV/c (`PrVeloUT`);
- SciFi hits matching the VELO-UT tracks are searched for within a window corresponding to a possible track transverse momentum above 1000 MeV/c and forward tracks are built (`PrForwardTracking`).

Other algorithms are run in order to decode raw data and prepare them to be used in the reconstruction algorithms. For the UT and SciFi detectors, the raw data already contains clusters, whereas VELO clusters need to be calculated from VELO hits. Clusters are then sorted in order to be used in the reconstruction algorithms mentioned above.

4.2 Performance tests

The throughput of the HLT1 sequence described in Sec. 4.1 is determined by using a dedicated machine with 20 physical cores, on which the sequence is run multiple times, each time by varying the number of reconstruction processes and the number of threads in each process. In this way, the maximum achievable throughput can be found. Each single algorithm is profiled by using standard tools such as Valgrind/Callgrind and Intel® vTune™.

The throughput is measured on a simulated sample of minimum bias events at $\sqrt{s} = 14$TeV and instantaneous luminosity of $2 \times 10^{33}\text{cm}^{-2}\text{s}^{-1}$. Events with fewer than 11000 total hits in the UT and SciFi are selected, as events with higher occupancy in these detectors cost significantly more in terms of reconstruction speed with no corresponding increase in signal efficiency.

The throughput of the sequence is shown in Fig. 4.2 as a function of the product of the number of processes and threads, for different values of the number of threads per process.

The maximum throughput of 12.4 kHz is found at 2 processes and 20 threads per process. A throughput of 3.4 kHz was measured in Ref. [6]. In that study, a sequence was used that did not include impact parameter cuts, and that required the transverse momentum thresholds to be above 500 MeV/c. This enables a wider physics reach that includes also multi-body beauty and charm decays, spectroscopy, electroweak processes and searches for exotic particles.

Fig. 4.3 shows a comparison of the throughput as a function of the impact parameter cut for various transverse momentum thresholds, thus probing different reconstruction and selection options (and therefore physics reach). An improvement of a factor of 2 with respect to the results of Ref. [6] is obtained. Fig. 4.3 also demonstrates that selections can be adjusted to further gain in throughput if necessary but with a cost on the physics reach.

Using the multi-threaded core framework gives a gain in throughput of around 20% with respect to a single-threaded framework (*non-Hive* line in Fig. 4.2). Fig. 4.4 gives the relative amount of time spent in each algorithm at peak throughput performance. Algorithms that perform data preparation (decoding, clustering and sorting of hits and clusters in the tracking subdetectors) account for about 50% of the overall time.

An example of the profiling of the algorithm performing the reconstruction of VELO tracks is shown in Fig. 4.5. In such profile plots, the time spent in each method of the algorithm is immediately visible, thus giving the basis for the algorithmic optimizations that result in the throughput improvement reported above.

4.3 Outlook

This chapter presented a first study in benchmarking the reconstruction part of the HLT1 application, with the purpose of showing the possible gains that can be achieved by optimizing algorithms in the Gaudi multi-threaded core software environment. The resulting throughput represents an improvement of at least a factor two over previous results, thus demonstrating the validity of the chosen approach. The throughput can be further increased by tightening reconstruction and selection thresholds, at the cost of shaping the physics program of the experiment.

In the displaced-track reconstruction sequence, the measured throughput is equally limited by the data preparation and reconstruction components of the sequence. A detailed profiling of the relevant algorithms highlights areas where improvements are necessary, thus guiding future development work.

The displaced-track reconstruction sequence presented here and described in Ref. [2] as a backup solution, is reasonably efficient for beauty and charm physics, still below the target but more efficient than in Run 2, due to the removal of the L0 hardware trigger whose fixed output rate of 1 MHz would otherwise saturate at the Run 3 luminosity. It is very inefficient for other physics processes.

This optimization work must continue in order to overcome these limitations and to cope with the main objective of triggering at 30 MHz. Detailed results will be regularly provided in status reports in the form of publicly accessible notes.

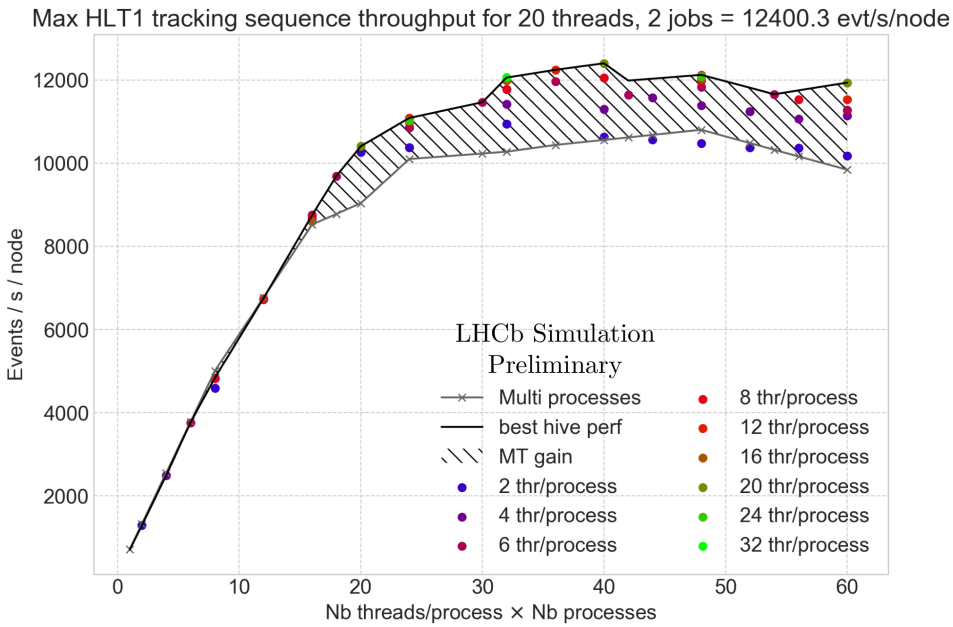


Figure 4.2: Throughput of the displaced-track reconstruction sequence, as a function of the product of the number of processes and number of threads, for different number of threads per process, as indicated in the legend. The throughput peak performance is 12400 evt/s/node for 2 processes and 20 threads per process. The "non-Hive" line indicates the performance that is achieved without multithreading.

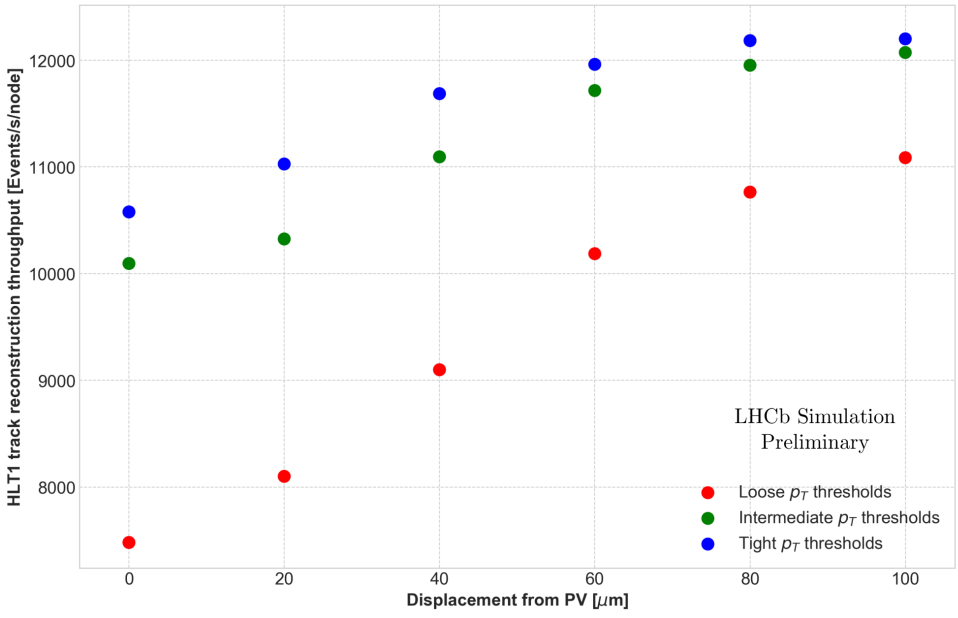


Figure 4.3: Maximum throughput of the displaced-track reconstruction sequence, as a function of the cut on the impact parameter (in μm), for different transverse momentum thresholds in the pattern recognition algorithms, as indicated in the legend.

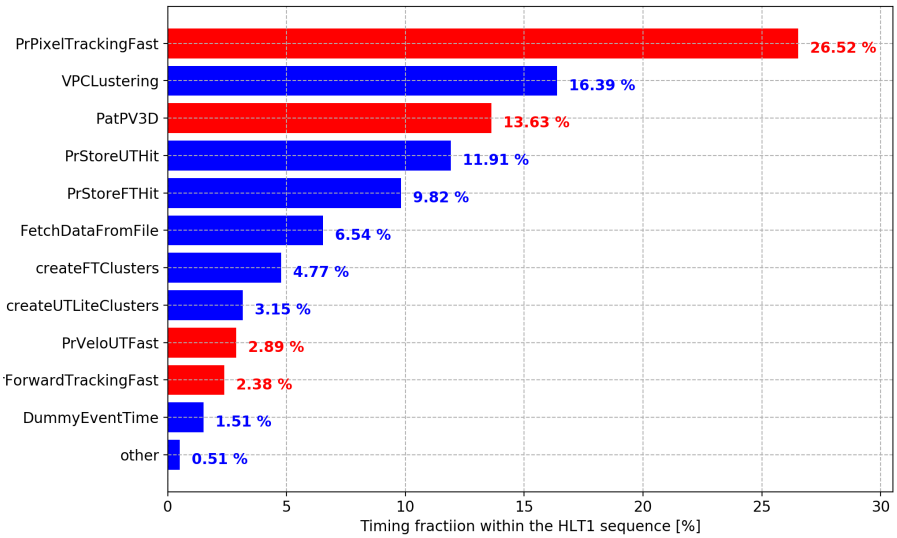


Figure 4.4: Timing division of the HLT1 sequence described in the text, for the peak throughput configuration. The timing fraction of the reconstruction algorithms is shown in red, while the timing fraction of data preparation algorithms is shown in blue.

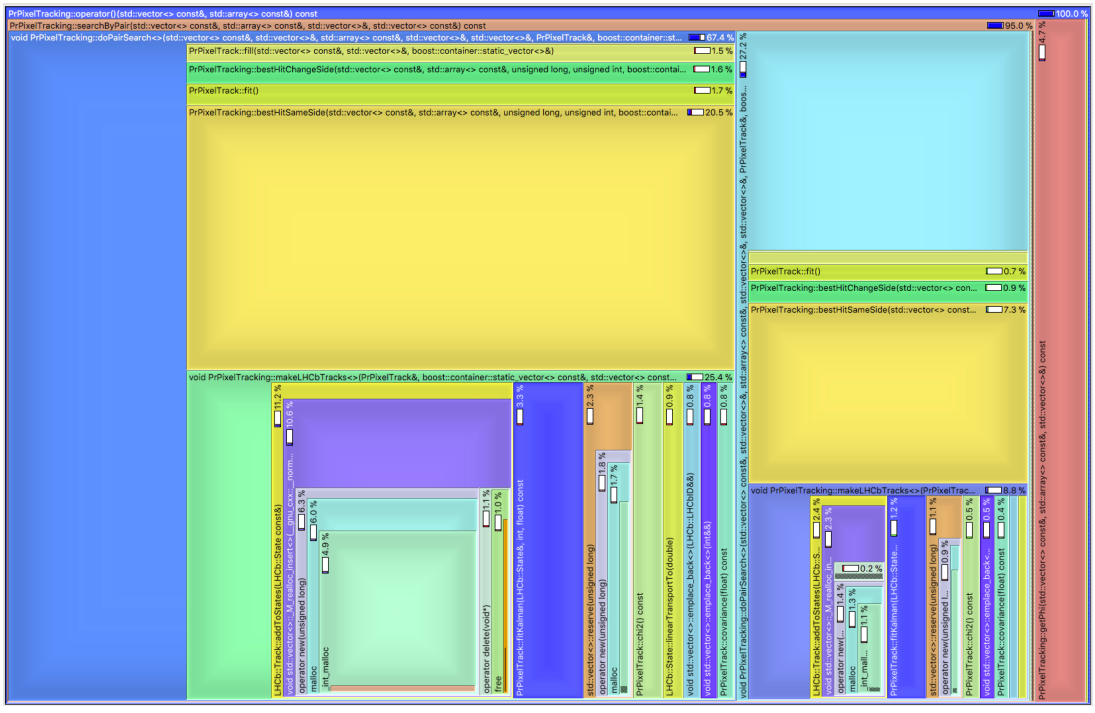


Figure 4.5: CALLGRIND profile (cycle estimation) of PrPixelTrackingFast algorithm. The area of each rectangle corresponds to the time spent in each method of the algorithm.

Chapter 5

Alternative hardware architectures

The baseline hardware architecture for LHCb is the x86 architecture, currently provided by Intel and AMD. In this section the possible use of alternative architectures or, in the case of hardware accelerators, also possibly complementary to x86, is discussed. Using accelerators or non-x86 CPU architectures has different implications so they will be discussed separately. Accelerators could also be used of course in conjunction with a non-x86 CPU.

Alternative architectures are particularly relevant for the HLT applications, where LHCb has full control and strives for the most efficient way to use the computing power that will be available, and that will be procured predominantly in late 2020 or even in early 2021 [2], depending on the final LHC schedule. Accelerators might give better performance for equal cost in the HLT and non-x86 CPUs can result in more competitive offers for the HLT servers. LHCb is less in control of resources outside of the HLT, *i.e.* on those provided by WLCG (Worldwide LHC Computing Grid) and elsewhere. However, also in this case it is useful to be able to use non-x86 resources, should they become available.

5.1 Non-x86 CPUs

The long dominance of the x86 microprocessor architecture in the data centre has eliminated practically all competing architectures from the desktop and server markets. In the last years however there has been increasing activity to develop alternatives, which can serve as drop-in replacements of x86 servers in a data centre. This means they look and behave like x86 servers, run the same OS and work with the same commodity auxiliary hardware such as hard-drives. Of course they require recompiled software[1].

The two dominant contenders are the OpenPower® architecture, a derivative of IBM's veteran POWER architecture and the ARMv8® architecture, which forms the basis of several different SoCs[2] targeted at the server market.

[1] In some cases emulators can be used, at the cost of a significant performance penalty. Moreover emulation works only for whole programs, which means that 'mixed' programs using proprietary libraries are not possible.

[2] SoC: system on chip, *i.e.* a CPU which includes most functions required by a fully functional server, I/O bridges, memory-controllers etc...

5.1.1 OpenPower®

The OpenPower® architecture is currently shipping the Power8+ CPU, which can provide up to 12 cores and up to 8 logical cores per physical core. The next generation that will be widely available in 2018 is the Power9 CPU, which in addition to higher core-counts has support for PCIe Gen4 and NVlink, enabling very fast connections to Nvidia GPGPUs. Power9 is the basis of two of the most ambitious upcoming SuperComputers in the United States in the framework of the CORAL initiative [39]. While the base-design is done by IBM the architecture is licensed to other vendors and IBM is building a large ecosystem of network vendors, platform designers and accelerator developers around it. Server platforms around Power9 are provided by several vendors, most of which also offer x86 systems. Contrary to their big-endian cousins used in legacy Unix workstations the OpenPower® CPUs run little-endian 64-bit Linux.

5.1.2 ARMv8®

The ARM® architecture is developed by the ARM company. ARM licenses the CPU architecture and/or instruction set to other companies such as Qualcomm, Apple, Cavium and Samsung which then design CPUs using it. ARM® is the dominant architecture in the embedded world, most mobile phones and tablets run on ARM®. Server processors differ from embedded CPUs in significant ways (number of memory channels, number of cores, floating point performance and many others) so it took a while until the first ARM® processors designed for use in servers became available. However, a significant number of major CPU producers are going to offer ARM® server CPUs in the immediate future. Like the OpenPower® machines, these come in the typical configuration of x86 servers, namely dual-socket machines with 6 to 8 memory channels and PCIe interfaces for the I/O. They run little-endian 64-bit Linux.

5.1.3 The case for OpenPower® and ARM®

Microsoft, Google, Amazon, Facebook and a couple of other similar companies are by far the most dominant influencers of the data centre technology.[3] These so-called hyperscalers are porting their software to OpenPower® and ARM® to improve their bargaining power when procuring CPUs.[4] LHCb is considering these architectures for the exact same reason. The CPUs account for about 30 to 40% of the total cost of a typical server. Competitive pricing will enable to get more CPU power for the same investment.

5.1.4 Program of work for OpenPower® and ARM®

The entire LHCb software stack needs to be ported to these architectures. LHCb does not use any closed-source libraries in its applications and both OpenPower® and ARM® run the same CENTOS distribution as the x86 CPUs. A priori the recompilation is straight-forward, excepting

[3] Just to put this in perspective: any single major data centre of any of the hyperscalers contains more servers than the entire LHC grid [40].

[4] Contrary to what can be sometimes read, power consumption plays no role in this, OpenPower® CPUs and ARM® server SoCs have no power advantage over Intel CPUs. This misconception probably is due to an "extrapolation" from the embedded SoCs found in mobile devices, where performance per watt is the key technology driver. In fact, Intel server CPUs are more power efficient than any current ARM® or OpenPower® competitor, reflecting the semi-conductor leadership of Intel.

for a few modifications in the build scripts. Nevertheless, this is not a trivial endeavour for the following reasons:

- Even though both architectures implement standard IEEE floating point numbers, there can still be differences in numerical results due to ambiguities in IEEE 754 [41].
- Vectorization which is one of the main advantages in exploiting modern CPUs is either done using architecture-specific "intrinsics" (essentially inline assembly instructions) or using wrapper libraries, with more or less efficient support for non x86 architectures. Vector operations significantly differ across CPU architectures, thus efficient code will require tailored vectorization or a truly optimized cross-platform abstraction library,

Apart from the error and warning-free compilation of the stack the following must be done:

- Define a test-set of quantities for validating numerical equivalence. Define criteria for numerical equivalence.[5]
- Choose an architecture-independent vectorization library and change vectorized code accordingly.
- Define automated unit-tests for the above.

The new architectures will be integrated in the standard nightly-build infrastructure. Once completed, a small test-batch of machines of either architecture will be acquired and put in production in the Online farm to gain long-term operational experience. Alternatively, if available, an equivalent cluster operated by the CERN IT-department could be used.

5.1.5 Example: Cross-Kalman project

As a motivation for the study of alternative architectures, Cross-Kalman is a good example. Cross-Kalman is a project to study the cost efficiency of various architectures using a highly portable, vectorized and efficient implementation of (parts of) the Kalman filter, which is one of the main CPU consumers in the HLT. Technical details can be found in Ref. [42].

Figure 5.1 shows the throughput of the Kalman filter across various architectures listed in the legend. In all cases the results were identical from a physics point of view, albeit with different run-times. The speed-up is normalized to a "standard" HLT node based on Xeon processors. In all cases measurements are taken on a fully loaded machine, with warm-up etc., to avoid distortions by turbo and hyperthreading.

Since very different chips are compared, the right graph in Figure 5.1 shows the relative cost advantage over the reference node, for those architectures were market prices were available.

5.2 Accelerators

Generically accelerators are devices that can accelerate a part of a complex computation, which is performed in conjunction with a standard CPU. In current data centres three main types can be found: General Purpose GPUs (GPGPUs), Intel XeonPhis and FPGAs. Only these will be considered in the following.

[5] There is no reason to assert that the numerical results on any one CPU architecture are a priori more "correct" than on another.

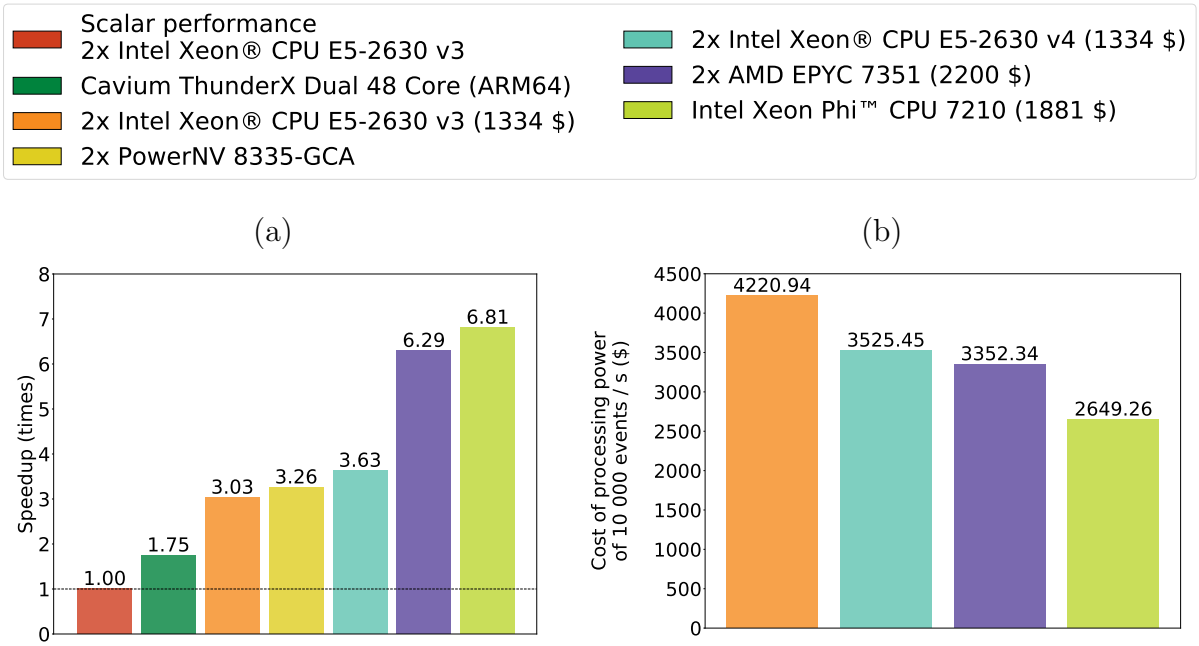


Figure 5.1: Cost using list-prices of running the Kalman filter on select architectures

5.2.1 Cost of using an accelerator

Using an accelerator does not come for free. The accelerator itself and the infrastructure required to run it must be costed. A formula for this is given below. The parameters are the following:

- cost impact on hosting server, e. This can be for example due to the fact that a 2U or 1U server is needed, where before a $\frac{1}{2}$-U unit was sufficient;
- CPU-cost for data-handling to/from the accelerator, d;
- usage efficiency of the accelerator, h. This is the fraction of the time of where either HLT1 or HLT2 are running and the accelerator is actually used;
- the cost of a CPU-node, c;
- the cost of an accelerator, a;
- the speed-up of the accelerator over the CPU (where CPU here means a full server), s, *i.e.* how many servers would be needed to replace a single unit of accelerator if the same operation done with the accelerator would be done with standard servers.

The proposed metric is performance per unit of cost and for the accelerator this evaluates to:

$$v = \frac{sh}{a + d + e}$$

Following this logic, proponents of an accelerator will have to demonstrate that:

$$v \ll \frac{1}{c}$$

A significant cost-advantage will also justify the additional person-power required to implement and maintain the accelerator. This effort will be depend strongly on the tools and libraries used.

5.2.2 Program of work for accelerators

A prototype implementation of a small-scale accelerator used for the calorimeter raw-data decoding has been proposed. This can be used as a template for similar use of accelerators before or "outside" the HLT.

The data-processing by the accelerator can in such a case be considered as part of the raw-data generation, *i.e.* it would be represented as it were another separate subdetector.

Such an accelerator code needs to be self-sufficient, *i.e.* it has to work with raw-data as input and has to add and/or transform raw-data as output. Since procurement of the bulk of the CPU power for the HLT farm will be done in 2020, at the latest at the beginning of 2021, a complete implementation proposal for any hardware accelerator, along with a convincing costing argument must be ready to be evaluated by the collaboration well in advance.

Chapter 6

Simulation

In order to generate simulated events, two applications are used in LHCb: Gauss, in charge of the event generation and particles tracking in the detector, and Boole, in charge of the simulation of the detector response. In the context of the software preparation for the LHCb upgrade the first of these applications poses special challenges.

Like all other LHCb applications, the simulation software is developed within the Gaudi core software framework. Therefore, many changes must be put in place for the Run 3 simulation to adapt to the multi-threaded Gaudi framework (Sec. 3.1), the new event model (Sec. 3.3) and Geometry Description (Sec. 3.5). On the other hand, it will be necessary to continue to produce simulated samples for Run 1 and Run 2 physics analysis during the next few years. As it will be explained in Sec. 6.6, the newer version of Gauss will support all data taking periods; in contrast, the newer version of Boole will be completely different from that supporting the current detector, since it will have to simulate the response of completely different subdetectors.

An additional challenge comes from the fact that parts of the simulation processing in Gauss are carried out by delegating the tasks to libraries and tool-kits developed in the high energy physics community at large, for example generators modelling the QCD and decay processes or Geant4 for the transport of particle through the material of the detector [7–9, 43, 44]. Special care is therefore needed in the use of these external components when they do not implement or have a different multi-threaded approach than that adopted in Gaudi.

Finally, the simulation is the biggest consumer of CPU, using about 80% of the distributed computing resources available to LHCb in Run 2. Furthermore, the increase in number of events that will need to be simulated to match the increase in luminosity and trigger rate in Run 3 will place an additional burden on the computing resources. To cope with these requirements, the speed of the simulation must be increased by at least an order of magnitude and possibly more.

This chapter reviews the various activities that need to be carried out in order to have a fully functional simulation for Run 3 that meets these demands and the work that has been initiated in this direction.

6.1 Computing requirements

Simulated samples are essential to design, build and commission the LHCb upgrade detector and the data processing applications. They are also fundamental for physics analysis and contribute to the precision of the LHCb physics measurements.

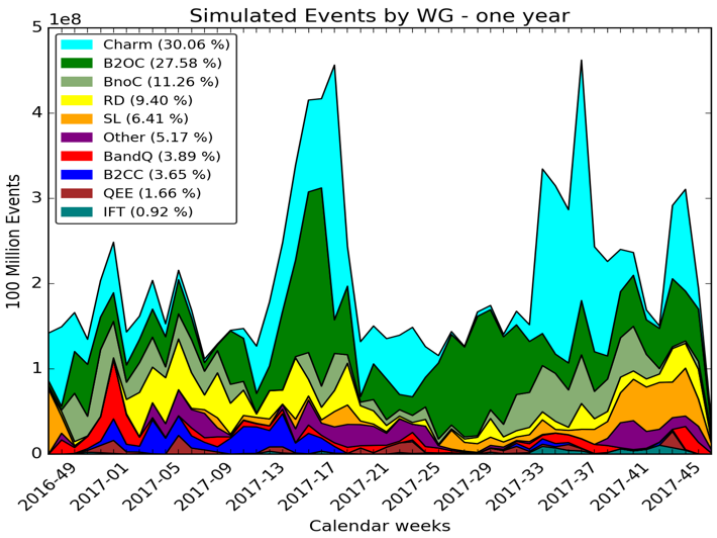


Figure 6.1: Number of simulated events per week in 2017, split by physics working group.

Requirements by Physics Working Groups (PWG) are strongly analysis-dependent and not easy to quantify, nevertheless in general one needs more simulated events for the signal of interest than those collected. For the first two LHC runs, LHCb has so far produced, reconstructed, recorded and analysed tens of billions of simulated events. As an example, Fig. 6.1 shows the number of simulated events as a function of time in 2017 on a weekly basis, categorised by physics working group. The total number of simulated events, about 10 billions, is twice the amount produced in 2016.

For Run 3 the number of events to be simulated will need to increase to match a factor five increase in luminosity. For fully hadronic channels, the trigger efficiency will increase by a factor two. This, in turn, implies also an increase in the number of simulated events needed to determine the analysis requirements after the trigger selection. In addition, with the full exploitation of the `Turbo` HLT paradigm deployed in Run 2 the purity of the events collected will be higher, with a corresponding growth in the simulated statistics required to satisfy analysis needs. It should be pointed out that for some analyses with large samples the amount of Monte Carlo (MC) events that can be produced is already insufficient and the uncertainty due to the limited amount of simulated samples is in many cases already dominant.

The simulation in Run 2 already accounts for about 80% of the distributed computing resources available to LHCb. In Run 3 with the reconstruction being done in real time the distributed computing resources may very well be used only by simulation and user analysis. In the last years the LHCb experiment has been looking into exploiting additional resources whenever possible. The possibility to gracefully stop the simulation application at the end of the current event has allowed the exploitation of opportunistic resources that might request fast draining of the worker nodes for prioritization reasons. A prime example of this is the usage of the Event Filter Farm up to its full capacity when not in use by the HLT, not only during the LHC (End of Year) Technical Stops but also during unscheduled stops and even between and during LHC fills. The use of other opportunistic resources (such as HPC centers, volunteer computing, or facilities not pledging resources to LHCb) for simulation is also being actively

explored (see Sec. 7.3).

Nevertheless, the computing budget that can be available by extrapolating the provisions of the last years will probably not allow to meet the growing needs of simulated samples with improved accuracy. In this scenario it is essential to optimise the simulation with the ideal goal of simulating all the events needed for the different analyses.

Memory allocation and computing execution time improvements expected from technological advancements or faster external libraries may be able to provide some speed-up by 2020, but a big factor will still remain to be gained. Therefore, an extensive use of faster alternatives is being actively pursued, under the assumption that it is possible to improve time performance without an unacceptable loss in physics accuracy.

The number of simulated events has also consequences on the storage needs. The LHCb experiment has already extensively tackled this problem with successive reductions in size of the simulated samples by using the following options:

- an optimization of the objects to be saved has been carried out at the beginning of Run 1 ensuring all information needed to be able to carry out physics performance studies would be present in the MC Particle/Vertices tree structure (MC truth) and by dropping the generators HepMC [45] event records;
- the packing of MC objects has been carried out together with that of the reconstructed event for simulated samples allowing a reduction in size by about a factor of 2;
- a new processing for MC samples has been introduced in production in 2013 where only interesting events are saved at the end of the processing chain (MC filtering, similar to real data stripping selection). While hundreds of millions of events need to be generated, transported through the modelling of the detector, triggered and reconstructed often only a small fraction satisfying certain conditions are of interested for a given physics analysis and hence has to be recorded. This has led in 2017 to a reduction by roughly a factor of two in the number of events recorded with respect to the number of events simulated;
- the μDSTs format has been introduced in production for the simulated samples in 2017. This format is the MC natural counterpart of the μDST for real data where, in addition to the reconstructed selected candidates, only the MC truth for the signal part of the event as associated to the selection of signal (online and offline) or the MC truth of the signal as generated is kept.

While the second option is always applied to the production of simulated samples, the third and the fourth are not suitable for every physics analysis, although their use is becoming more widespread. Due to the `Turbo` HLT paradigm it is expected that the majority of production of simulation samples for Run 3 will consist of MC-filtered μDST samples. It is also intended to eliminate the few rare cases where the first option is not applicable by appropriate modifications of the MC event model.

6.2 Simulation in the core software

In the introduction of this chapter it was mentioned that the simulation like all other LHCb application will need to adapt to the new Gaudi components. In particular a re-factoring of

many simulation-specific modules is necessary to make them compliant with the requirements of thread-safety and re-entrance. Another significant change will be the adaptation to the constraints of the new event model, especially the read-only Transient Event Store.

6.2.1 Gauss and Gaussino

In the evolution of Gauss to a new thread-safe framework based on the Gaudi task scheduler, its underlying modular and configurable design will be exploited to the fullest. In this context a collaboration was started with the SFT group at CERN to factorize the experiment-independent components to provide all basic functionalities into a separate framework, called Gaussino. This Gaudi-based, experiment-independent simulation infrastructure should provide the steering of the generators essentially using the design and implementation of Gauss as-is, interfaces to configure and steer Geant4, transparent ways of passing the generator event to Geant4 [7–9] and to retrieve the outcome of the particle transport from Geant4. Possibilities to interface other software than Geant4 for the detector simulation should also be supported. The intention is that the future version of Gauss will be based on this framework and provide the LHCb-specific functionality either via dedicated configurations of the Gaussino components or specific software implementing the common interfaces, for example in case of additional generators. A sketch of the dependencies of the current and future versions Gauss from the underlying projects and libraries is given in Fig. 6.2.

Gaussino should allow an easier way to implement the new features of the task-based Gaudi framework and test them with simpler settings but in an environment very similar to that of LHCb. A first version of Gauss based on Gaussino and supporting only the default LHCb simulation configuration will allow to test the MC event model and new geometry format and allow to explore more efficiently the optimal way to parallelize the simulation (Sec. 6.2.4). Once the choices are implemented in the reduced Gaussino environment and validated with a minimal Gauss application, they can be propagated to the rest of the Gauss components, *i.e.* all generator adaptors and fast simulations.

As mentioned earlier the basic architecture of Gauss will be kept in Gaussino. Gauss is structured in two main independent phases (event generation and detector simulation) that can be run together or separately. Normally they are run in sequence as a single job since the first phase is much faster than the second one. A schematic view of the application phases is shown in Fig. 6.3.

The first phase consists of the event generation of *pp* collisions and the decay of particles in the channels of interest for the LHCb physics program. The generation phase itself is structured as an algorithm that delegates specific tasks to dedicated tools implementing well defined interfaces, like fore example number of collisions per bunch crossing, the setting of the beam parameters for the collision, the production of a given collision, the decay of particles after the hadronization, the selection of channels of interest, and smearing of the position of the collision [46]. This modular design has proven to be very flexible as it has allowed integrating new generators by providing suitable adaptors that can then be used with the Gaudi "plug-and-play" mechanism at run time. The heavy ion and fixed target program of LHCb in recent years is a good example of how Gauss could provide the required simulation samples with minimal effort since as soon as new tools for the setting of the beam parameters (*e.g.* fixed target) and a new dedicated heavy ion generator for the collision were implemented they could be used without modifying any other code in the application. Different generators can be used in the same event (*i.e.* bunch crossing)

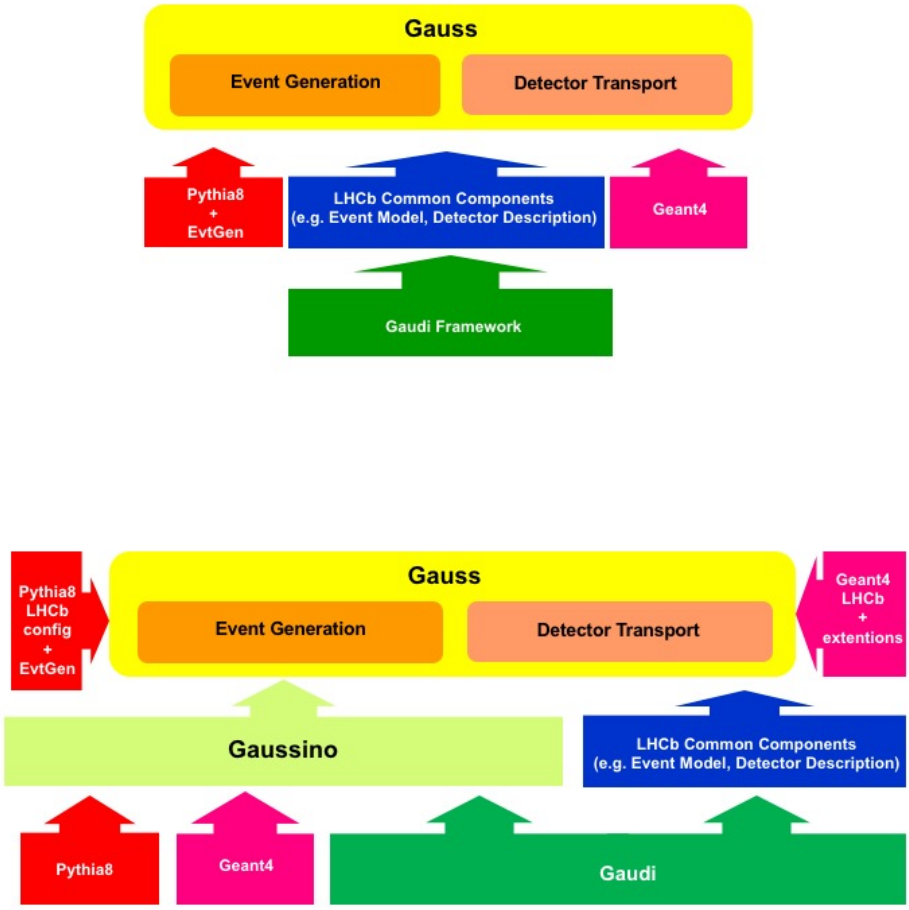


Figure 6.2: (top) Dependencies from LHCb and external software for the current Gauss (bottom) Dependecies from LHCb, Gaussino and external software for the future Gauss

for each of the concurrent collisions (pileup) and to handle the production and decay of particles. Therefore, the main work needed for the generator phase is a repackaging of the different parts to take away any LHCb-specific parts from Gaussino and encapsulate them in the future Gauss version.

The second phase of Gauss consists in the tracking through the LHCb detector of the particles produced by the generator phase. The simulation of the physics processes, which the particles undergo when travelling through the experimental setup, is currently delegated to the Geant4 toolkit. Geant4 interacts with Gauss using a set of interfaces and encapsulated converters in a Gaudi specialised framework (`GiGa` [47]). In the current implementation of Gauss the main simulation algorithm is strictly tied to the use of the Geant4 tool-kit for the

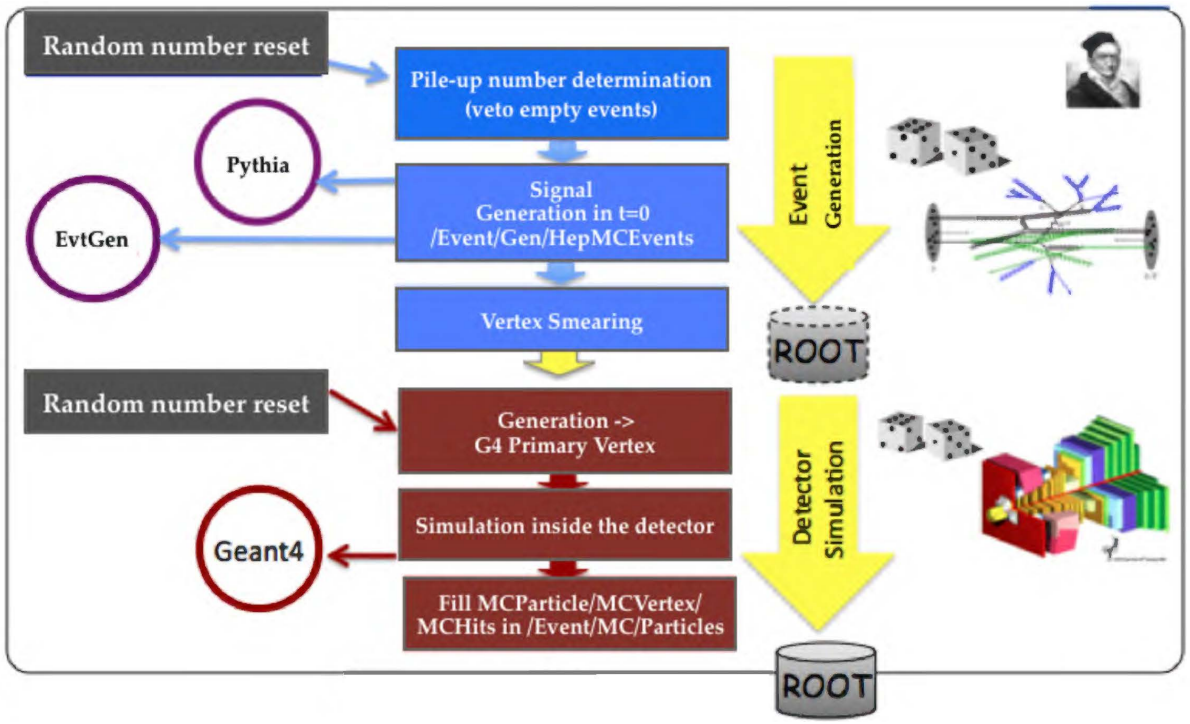


Figure 6.3: Gauss event processing sequence

whole event processing and the use of converters makes it difficult to adopt the multi-threaded version of Geant4 (Geant4-MT). A full redesign of the interfaces to various components of the simulation (passing the geometry, configuring the physics processes, providing the particles to be transported and retrieving the results after triggering the transport), will be necessary in view of making them more experiment-independent as well as adapting the components to Geant4-MT as well as other simulation engines. The future Gauss is in fact intended to evolve into an integrated simulation framework fully exploiting the potential of its architecture: the full simulation phase can already be replaced with a different algorithm and this approach has been proven in the implementation of a fully parametric solution currently being worked on. Furthermore, a prototype of an extension of the Gauss application, where Geant4 is replaced by dedicated fast simulations of either only selected detector parts or particle species, has been recently implemented in the context of the effort to speed up the simulation. The fast simulation options that are being developed are described in Sec. 6.3.

6.2.2 Event generators

The set of generators integrated in Gauss via dedicated adaptors is constantly increasing to match the widening of the physics program of the experiment. The majority of them are well established libraries, usually provided by groups of theorists, that are available to the high energy physics community at large (*e.g.* Pythia [43, 44], EPOS [48]) while a few others have been developed and are maintained by the experimental community (*e.g.* EvtGen [49]).

Support and maintenance need to continue to be provided for the adaptors implemented in

Gauss. The implementation of new adaptors may become necessary as well. Problems may arise when moving the generator phase of Gauss to a parallelized processing, as the external libraries may or may not be thread-safe, In particular, special care will need to be taken for legacy generator libraries implemented in FORTRAN. The parallelization of the generator phase is not mandatory, and an hybrid solution where multiple events are generated sequentially and the events are later transported in the detector in parallel in Gaussino may be considered since the generation phase is normally much faster than the detector simulation. The decision of having or not an hybrid system will be made based on how the Gaudi task and event scheduler will be used in Gaussino and Gauss.

The implementation of new generators will continue to be carried out in the current version of Gauss until its future version will provide equivalent functionality for producing the majority of the samples.

6.2.3 HepMC3 and Monte Carlo event model

HepMC [45] is the de-facto standard event generator exchange format in the high energy physics community. All modern generators provide the events in this format to the outside world. HepMC is used in Gauss for transferring the information between different generators tasks as well as for passing relevant particles to Geant4. Additionally, it is used while transporting the particles in Geant4 to store the history of the physics processes that take place in the detector.

A major update of HepMC, HepMC3, was initiated in 2014 and a new version has been made available at the beginning of 2017. This new version ensures thread-safety and a smaller memory footprint; it also implements new features required mainly by generators' authors, for example a clear separation of core records and utilities, more standardisation and integration with Les Houches Event Files [50].

In the future version of Gauss, HepMC will be used in memory but it will not be provided within the persistent LHCb event model. All necessary information needed for physics analysis will be transferred from HepMC to the `MCParticles` and `MCVertices` structures.

Some of the issues that need to be addressed for simulated objects have already been discussed in Sec. 3.3. The immutability of the new event model poses particular constraints on how the simulation has to transfer data among its separate phases as a simulated particle may be "modified" while being transported in the detector because of a particular physics process producing new particles. The solution envisioned is to fully exploit the extensibility of data and continuous iterations over multiple Transient Event Store locations in Gauss. Extensibility of data will also allow to unify simulated and reconstructed particles into a common class to be extended with MC truth information in one case and reconstructed parameters in the other. While continuous iteration will be essential in future Gauss the consolidation of the MC truth containers, like `MCParticles`, when they are persistified is an option that will be explored to ease the data analysis in the subsequent applications. The relation tree of MC truth between `MCParticles` and `MCVertices` needs to be accessed in both directions. The proposed solution is that the relations will be created once and stored. While in the current event model the relation between `MCHits` and `MCParticles` is only in the direction of the hits to the particles the create-once-and-store solution would also be applicable for `MCParticles` to `MCHits` relations allowing to ease the data analysis for detector performance evaluation.

The choice between AoS or SoA as format container for the MC event model will be made after evaluation of the use and benchmarking a prototype implemetation of the SoA option in

the GAUSSINO framework.

6.2.4 GEANT4 and GAUDI: multi-process, multi-threading and vectorisation

The simulation applications will run within the multi-thread version of the GAUDI framework. This poses a compatibility issue with the multi-thread version of external tools like GEANT4. Integration tests have been done within ATLAS, where design choices have been made to make the different concurrent models of GAUDI and GEANT4 work together. Similar choices will be adopted and tested in GAUSSINO. Nevertheless, all the LHCb simulation-specific code will need to be adapted. This and further evolutions need to be fully tested. Of course, the effective improvement given by the parallelization will strongly depend on the choices for multi-thread/process jobs on the grid. The parallelization requires the migration to GEANT4V10, completed in 2017, and the integration of concurrent models. The possibility of exploiting the GEANT4 multi-threaded capability for parallel processing of other collisions in the same and in adjacent bunch crossings will need to be implemented and thoroughly tested to allow its evaluation after benchmarking measurements. In-depth measurements and validation of the option already available in the current GAUSS to process events in parallel on different CPU processors will also need to be made.

6.2.5 Geometry for simulation

In any simulation with complicated geometry, the majority of time is spent navigating through the geometry itself. The change in the detector description (see Sec. 3.5) will have an impact on how GAUSS transfers the geometry information to GEANT4. A new way of passing the geometry to GEANT4, replacing the `GiGa` converters [47] with a dedicated geometry service, has been implemented in 2017 to prepare for this as well as to ease the migration to the multi-threaded version of GEANT4.

New geometry packages with improved performances have become available for GEANT4, like `USolid` and `VecGeom` [34]. The latter has been developed in the context of GEANT4V [51] and provides both a scalar and a fully vectorized option. These alternative versions of GEANT4 geometry have been tested in the GAUSS framework: no improvement was observed when using `USolid` and preliminary studies do not show an improvement when `VecGeom` scalar is used, likely due to the type of shapes used in LHCb. Some improvements are expected with the vectorised version of `VecGeom` that is going to be tried out as soon as a GEANT4 multi-threaded version that can fully exploit it is functional in GAUSS.

A simplified geometry description, either via utilities provided by GEANT4 or the LHCb detector description should be evaluated in the context of its use for some of the fast simulation options as it would ensure automatic consistency between the detector description used by the various simulation options.

6.3 Fast simulation options

Due to the increase in size of requested MC samples in the last years, the flexibility of the GAUSS simulation framework was used to full advantage by implementing a variety of alternative configurations. Many options are possible: simulating only part of the event, replacing a full GEANT4 simulation with faster versions for a given detector (fully parametric, based on a database

library or faster GEANT4 implementations) or disabling specific processes (*e.g.* Cherenkov effect in RICH detectors), stopping particles at a specific stage, simplifying the geometry (*e.g.* removing sub-detectors), re-using the underlying event or merging a simulated signal with real data, or even using a fully-parametric simulation (using as much as possible available packages like DELPHES [52]).

Several solutions are being developed in parallel and, as the palette of simulation options increases, it must be ensured that predefined safe and easy-to-use configurations are provided to the collaboration. Particular care has to be taken in the implementation of GAUSSINO and future GAUSS, see Sec. 6.2.1, as fully integrated simulation frameworks to achieve this. Effort will be needed to make these fast simulation options easily accessible to the final user and in the LHCb production environment. The most appropriate configuration will then be recommended for each given analysis, for example balancing between a faster processing time and a lower accuracy for specific particle types.

Some of the possibilities listed above are already implemented in GAUSS, allowing to explore their impact on the physics analysis. They have been deployed for Run 1 and Run 2 production as they became mature. Although all options could in principle be explored, those that would be useful for many physics analyses and/or would give a bigger decrease on processing time were exploited first.

6.3.1 Re-decay

The reuse of the non signal part of an event multiple times by combining it once simulated with N simulations of signal [53], where N is of the order of 100, has been shown to be appropriate for some analysis and is currently used in production with a CPU gain of a factor between 10 and 20 depending on the multiplicity of the decay of interest. The increase in number of events produced is clearly visible in Fig. 6.1 where the peak in the number of events for the Charm WG are due to productions making use of this option. On the other hand, the option of merging a simulated signal with real data is not currently planned as it implies considerable constraints both in term of physics consistency and production infrastructure.

6.3.2 Fast calorimeter simulation – shower libraries

The existing simulation infrastructure was extended to allow for the choice of alternative options for specific detectors not only at the level of full events but also single particles. This is providing the basis to inject new fast implementation for the simulation of the calorimeters. Replacing the GEANT4 simulation using this interface to provide calorimeter MC hits can give a gain up of up to almost a factor of 2 in the overall simulation time. Concrete implementations of calorimeter showers are being developed using both a classical "frozen showers library" approach as used in ATLAS [54], and a new Machine Learning method based on Generative Adversarial Networks [55].

6.3.3 Parametric simulation

A fully-parametric solution using DELPHES is also under implementation. It was demonstrated it could be used in a functional system for analysis by providing the same type of minimal reconstructed objects (*i.e.* tracks) that the analysis requires. Work has now started to provide all types of information needed, from tracks to calorimeter objects and to RICH particle ID

information (including their uncertainties). Generating the necessary efficiency and resolutions for different data-taking periods and providing them in a straight forward way is essential in order to fully exploit the use of the DELPHES-based parametrization in GAUSS for physics analysis. Providing expected performance values will also allow to evaluate the physics reach of possible future upgrade of the LHCb detector.

The possible re-use of the RICH particle identification fast simulation solution that is being explored for DELPHES is also considered as a stand-alone fast simulation for the RICH as it would allow to save almost another factor of two in the overall simulation time. Nevertheless, some modifications in the configuration of the subsequent reconstruction application will be necessary so that it can produce the particle identification information from the modified GAUSS output and produce the final samples with all the standard information.

6.4 Code and algorithm optimization

In addition to providing alternative faster solutions, the simulation software itself needs to be made faster, and an in-depth code optimization should be carried out. Therefore, detailed timing measurements of GAUSS were made in view of reviewing some of the simulation settings, where more fine-grained solutions may be appropriate. In addition, an evaluation of the optimal cuts for each detector region and particle type will be carried out to obtain an improved computing performance with the same or better physics modelling. One year is needed for this optimization and review of configuration settings. Alternative modelling of some physics process should also be investigated in terms of reliability of the description against time spent.

As an example, a timing test was run using the LHCb Performance and Regression timing service (LHCbPR, [56], see also Sec. 8.2.2) before and after implementing an optimization of the simulation of the physics processes in the RICH detectors. An overall gain in execution time of about 33% was observed.

6.5 Monte Carlo truth

In Monte Carlo workflow, the MOORE application will run on the output of the digitisation produced by BOOLE to produce a reconstruction output in the same format as in the real data streams. The application should propagate to the output a subset of the MC truth available from BOOLE, but also produce and save MC truth of its own (in particular the relationships between reconstructed particles, tracks, vertices and the `MCParticles` and `MCVertices` that originated them). This can be done either as a final stage of the MOORE event processing, or in a separate application. The latter case is already implemented for the `Turbo` stream in TESLA.

6.6 Simulation for Run 1 and Run 2 analyses

Simulated samples for physics analysis will need to be provided for Run 1 and Run 2 in parallel with simulation for Run 3 analyses. In fact the vast majority of the samples to be produced in the next few years is expected to be for analysis of Run 1 and Run 2 data, at least until a sizeable amount of data is collected with the upgraded detector.[1]

[1] A notable exception is the production of the sample of minimum bias events in upgrade conditions, needed for the commissioning of the entire upgrade data flow.

In this perspective, rather than supporting two separate simulation applications or using a frozen version of the simulation for Run 1 and Run 2, the decision has been taken that future versions of Gauss should be able to produce simulated samples for all data taking periods, in order to profit from the continuous evolution in accuracy of the physics modelling expected from HEP generators and Geant4, largely due to inputs from LHC physics results. On the other hand, as mentioned at the beginning of this chapter, Boole will be completely rewritten to provide the digitization of the new detectors.

Therefore, the future versions of Gauss will also have to support the current detector geometry. Different solutions are being considered which require different levels of effort for implementation and maintenance. The current plan is to migrate only the upgrade geometry to the new format (Sec. 3.5.2). As a result, the future Gauss will have also to be able to load the current detector geometry and simulation conditions. This implies that either the current geometry code and associated services will have to be maintained until the simulation support for the current detector is no longer necessary or the current detector geometry is loaded via the GDML exchange format and an alternative way to load the simulation conditions is found.

A second strong implication of this choice is that while the future Gauss will use in memory only the new event model, a mechanism to save MC data objects ina format compatible with Run 1 and Run 2 software applications must be provided. Hence converters or dedicated packing must be foreseen in Gauss to support data persistance in the current event model. Alternatively, a dedicated conversion application to be executed between Gauss and Boole when producing samples for Run 1 and Run 2 could be envisaged. In case backward-incompatible changes would need to be introduced for added functionality the need of dedicated patches to the Run 1 and Run 2 applications may arise. Some experience has already been gained with this approach with the current production where patches of Run 1 software were necessary.

Chapter 7

Distributed computing and analysis model

The distributed computing of LHCb is based on the DIRAC interware, and its LHCbDIRAC extension. The DIRAC project is developing interware to build and operate distributed computing systems. It provides a development framework and a rich set of services for both Workload and data management tasks of large scientific communities.

7.1 DIRAC history and context

DIRAC [57] was started by LHCb as project for accessing Grid resources. In 2009, following interest of other communities, the project has been open sourced. Now, it is a truly open source project hosted on GitHub and released under the GPLv3 license. Within the following years, DIRAC has been adopted by several experimental communities both inside and outside high energy physics, with different goals, intents, resources and workflows.

There is no dedicated funding for its development: instead, communities using it are welcome to participate in its development. DIRAC is publicly documented, with an active assistance forum animated by the users and developers themselves. A yearly user workshop and weekly open developers meetings gather users and experts. The project counts about five core developers, and a dozen contributing developers, with a dominant role of LHCb ones.

The DIRAC consortium has been established in 2014 as a representing body for the development and maintenance of the DIRAC software. The consortium counts a small set of active members, each of which elects a representative, while consortium members elect a director, and a technical director. Both the director and the technical director roles are covered by members of LHCb. Institutes that are part of the consortium engage in the maintenance and in the promotion of the DIRAC software.

7.2 DIRAC key features: extensibility and flexibility

Communities that have chosen DIRAC as the distributed system of choice vary greatly: some of them have hundreds of users, others only a few. A community using DIRAC may run concurrently several tens of thousands of processes ("jobs") on grids and clouds, and store and analyze petabytes of data on remote storage; others only run few hundreds of jobs per week, and

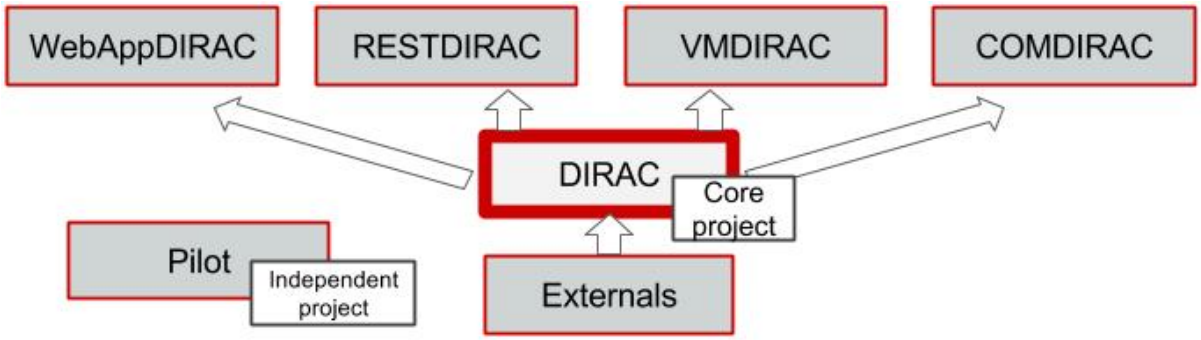


Figure 7.1: Horizontal extensibility in DIRAC

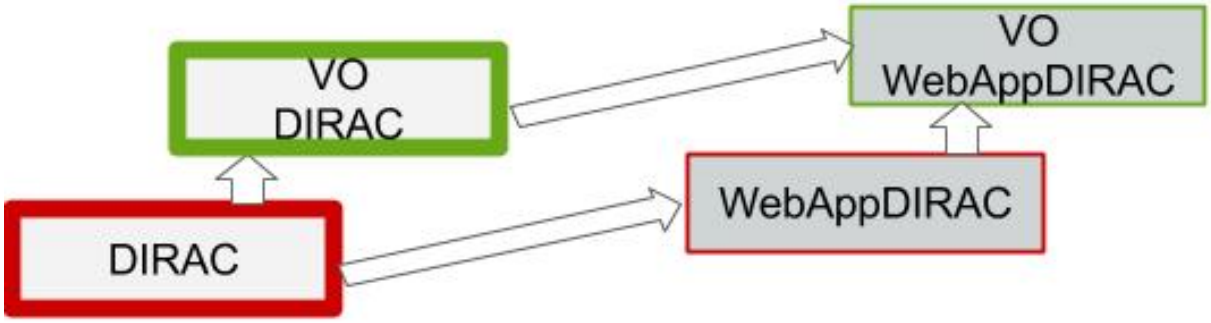


Figure 7.2: DIRAC vertical extensibility

barely need to interact with storage systems. Requirements vary, and DIRAC should be able to accommodate most of them.

In order to accommodate different requirements, DIRAC has been developed with extensibility in mind. Two types of extensibility are recognized, that are dubbed "horizontal" and "vertical". Fig. 7.1 shows a diagram that explains the so-called "horizontal extensibility": the core project ("DIRAC"), can be extended for accommodating different requirements. Each of the projects in Fig. 7.1 has been created with a specific goal, and each is independently versioned. At the same time a DIRAC release is composed by all those projects, so there is a strong dependency between them. All the projects in Fig. 7.1 are hosted together on GitHub, share the same license, and are maintained by the same set of users.

The "DIRAC" software is the *core project*, so it is the glue that keeps the satellite projects together. It also depends on software not maintained by the DIRAC consortium, which is collected in "Externals". The code for the DIRAC web portal [58] sits in WEBAPPDIRAC. RESTDIRAC [59] extends some DIRAC services by providing a REST (Representational State Transfer) interface. With VMDIRAC [60] it is possible to create Virtual Machines on several clouds. COMDIRAC extends the DIRAC user interface. The Pilot [61] is instead independent from all the other code.

Fig. 7.2 shows a graphical representation of vertical extensibility, that enables DIRAC users and Virtual Organizations (VOs) to extend the functionalities of the basic projects. In other words, the projects of Figure 7.2 can be extended by the communities in order to provide specific functionalities.

Coding for DIRAC also requires developing with flexibility in mind. There are several examples of DIRAC's flexibility, and one example regards DIRAC not imposing any computing model on its users. For instance, a tiered structure of computing centres may not mean anything to certain VOs. DIRAC provides users the possibility for a "full mesh" computing model, where every job can, in principle, run everywhere. At the same time, a "real world" computing model is represented by putting limits on a full mesh, and this is simply a configuration parameter.

7.2.1 DIRAC for LHCb

LHCb has implemented an extension of DIRAC (LHCbDIRAC [62]) and uses fully the functionalities that DIRAC provides. LHCb has customized some of the DIRAC systems, and created two new systems: the Bookkeeping [63] and the Production Management [64]. They both also provide GUI extensions within the LHCbWEBAPPDIRAC (the LHCb extension of WEBAPPDIRAC). The DIRACPILOT project is extended within the LHCBPILOT project.

7.3 Exploiting computing resources

In the last few years, new types of computing models, such as IaaS (Infrastructure as a Service) and IaaC (Infrastructure as a Client), gained popularity. New resources may come as part of pledged resources, while others are opportunistic. Most but not all of these new infrastructures are based on virtualization techniques. In addition, some of them present opportunities for multi-processor computing slots to the users. Virtual Organizations are therefore facing heterogeneity of the available resources and the use of a software framework like DIRAC to provide the transparent, uniform interface has become essential. The transparent access to the underlying resources is realized by implementing the pilot model. DIRAC's generic pilots [65] (the so-called PILOTS 2.0) use a plugin mechanism that makes them easily adaptable. PILOTS 2.0 have been used for fetching and running jobs on every type of resource, being it a Worker Node (WN), behind a CREAM [66] or ARC [67] or HTCondor [68] or DIRAC "SSH" Computing elements, or a Virtual Machine running on IaaC infrastructures like Vac [69] or BOINC [70], on IaaS cloud resources managed by Vcycle, or the LHCb High Level Trigger farm nodes, or any type of opportunistic computing resource.

While for LHC VOs the WLCG resources still provide the majority of available computing power, this is certainly not true for non-LHC communities. Nowadays DIRAC is used by dozens of VOs with a variety of computing models and resources among them.

In other terms, the Grid "push" model is still in use, but is complemented by IaaS and IaaC models. A typical case of IaaS is represented by cloud systems, while vacuum models [69] can be considered as embracing the IaaC model. Traditional grid sites are also presenting heterogeneous resources due to multiple hardware generations being supported. The grid is not anymore "The Grid", as the distributed systems of present-day experiments are made of heterogeneous resources.

The bulk of changes done for PILOTS 2.0 serve as a base for the new generation of DIRAC pilots (PILOTS 3), that include more general-purpose pilot commands, but also higher monitoring capabilities. PILOTS 3 are already used in production within a subset of resources, using the exposure control deployment strategy.

The scalable monitoring system based on the Message Queue (MQ) architecture is foreseen to be included as an option to the Pilot code. The monitoring system running as early as the pilot scripts itself, before any job matching, will be able to provide information about the WN

environment and possible errors occurring during the installation and configuration phases, by sending logging messages to the dedicated servers.

7.4 DIRAC scalability

The DIRAC plans for increasing the scalability of the overall system take in consideration that the main requirement is keeping a running system working, with continuity. This requirement translates into the need of studies and developments within the current DIRAC framework in order to check the system scalability in terms of

- traffic growth, meaning increase of rate;
- dataset growth, meaning increase of data;
- maintainability.

For what regards traffic growth, the first evaluation is about the DIRAC architecture. The main question is if it is time or not for a complete re-engineering, involving architectural changes. Looking at differences between modern architectures and the current DIRAC architecture, it was noticed that the latter, even if designed about 10 years ago, still does not look very different from what modern theory suggests. There are anyway changes to the technology that intervened during these years, to which DIRAC only partially reacted, and which will need to be taken into account. Some of these technologies include, but are not limited to, Message Queues, centralized logging systems, multiprocessing wherever possible, and the move to Python 3. Moreover, the DIRAC `Diset` communication protocol, and its main dependency which is the `pyGSI` package, can be largely replaced by moving to Python 3. For what regards purely the dependency from `pyGSI`, it is going to be removed already before moving the whole framework to Python 3. A proof of concept about running DIRAC on an orchestrator [71] was also made.

Assuring full scalability for the dataset growth means mostly determining if the databases DIRAC is currently working with can sustain an increased load. While relational databases do not pose, generically, a scalability threat for structured data, NoSQL implementations can become useful for handling unstructured data like monitoring data, and some preliminary tests already show good results. Object and block storages should also be evaluated for static unstructured data like logs and sandboxes.

System and software maintainability have been also recognized as a critical point for assuring scalability. A maintainable system needs proper monitoring, easy administration of components and databases. But it also means easily maintainable code, with increasing functionality tests, the use of continuous integration tools and performance tests, better documentation and user support. All these points are already part of the constant evolution of DIRAC.

The conclusion is that new technologies must be integrated, and some core part of DIRAC must be replaced, but overall only evolutions are needed, not a complete redesign. Few points have been identified in which the new technologies can be used in order to cope with the increase of data and job rate.

7.4.1 Scaling the DIRAC workload management

The DIRAC Workload Management System (WMS) is the key component for handling jobs with DIRAC. The "task queue" database is, as of today, a representation of the job queues within the

DIRAC WMS databases. Today's implementation uses a relational schema implemented within MySQL DB. The use of proper message queueing technologies will be considered. Unstructured data like job parameters and job attributes have been, up to now, persisted within relational databases, with limited extensibility and limited historical perspective. The persistence of these data in NoSQL databases is thought to be a solution to the above, and its implementation will start already in Run 2.

7.4.2 Scaling the DIRAC data management

The DIRAC Data Management System (DMS) is the entry door for managing LHCb distributed storage resources and data. Its main relational database, the "DIRAC File Catalog", has been re-engineered few years ago, and for the occasion it was demonstrated that it could handle ten times the data it currently handles, which should give us enough safety margin going into the LHCb upgrade.

7.4.3 Scaling the LHCb bookkeeping

The LHCb bookkeeping is a metadata and provenance catalogue used to record information about the datasets such as creation, transformation, type and quality of the data. One of the key components of the system is the LHCb Bookkeeping Database (BKDB) which is persisted in the Oracle relational database management system. In order to cope with the rapidly increasing data size, the main tables and indexes are partitioned. This allows to run more efficient queries using, for example, partition wise joins. Ongoing effort is going in using the In-Memory Column Store and into dropping materialized views.

7.5 User analysis

Everything that is done for LHCbDIRAC will also benefit user analysis. The GANGA tool [72] was the software of choice for LHCb users-driven analysis jobs. Software maintenance, and continuous users feedback is needed for adapting to LHCb users needs in the upgrade era, and possible changes in DIRAC APIs and in the LHCb core software. Working Group productions, discussed in Sec. 2.4.3, are encouraged as a good approach for doing analysis on the Grid where requests, coming from the working groups, are managed by the central production team. Working Group productions do not eliminate fully the need for private analysis jobs on the Grid, but are thought as a better organized way of running large working group analysis.

Individual users will still need to submit jobs to *e.g.* test their code before centralised production, or even to launch their private production (even though the latter will be unpractical, due to the expected size of the datasets). The GANGA tool was used in Run 2 for that. It will be used in Run 3 as well. The possibility of slimming down GANGA by keeping only the features relevant to LHCb should be explored, in order to overcome scalability bottlenecks that might appear in case of large user productions.

Chapter 8

Software infrastructure

8.1 External software dependencies

The LHCb software stack depends on many software packages developed at CERN, in other scientific institutions or in industry. Many of these packages are Open Source. The experiment therefore needs a way to track which external packages (and which version) are needed, as well as a way to compile and distribute them.

This is a common need among the LHC experiments. Furthermore, many of the external packages are used by all LHC experiments. The EP-SFT group at CERN prepares this software as common "LCG releases" [73]. The LHCb experiment currently uses in the order of 70 external software packages from the LCG releases

They are prepared using the LCGCmake [74] tool. This tool requires CMake, the compiler needed for the release and few system dependencies (the HEP_OSlibs [75] RPM can be used on CentOS7 and Scientific Linux CERN version 6 to install the system packages needed).

One important aspect of the LCG releases is that they do not have to rely on the system compiler but can use a more recent version. This is required by the LHC experiments in order to use recent C++ standards and profit in the latest improvements in term of execution speed of the compiled programs.

The LCG releases are available for a limited number of operating systems and the production system for LHCb is the one currently used in the LHC computing grid (CentOS7).

8.2 Infrastructure

As many developers are involved in the development of the millions of lines needed for the experiment framework, a strong version control system and good processes are crucial to ensure the quality of the software.

The LHCb core software team strives to continually improve the software management method following the evolutions of the software industry, and this is expected to continue in a similar fashion after the LHCb upgrade.

In the past, many tools were written within the experiment to deal with the HEP specific environment.Current tools tend to match LHCb's requirements, therefore they are being standardized and simplified as much as possible (*e.g.* moving from CMT to CMake, using the JENKINS continuous integration system to drive the nightly builds, packaging with RPMs etc...).

8.2.1 Code organization, source code management and version control system

The LHCb code base is split into several projects, versioned and managed independently, each having a distinct goal. Each of these projects is managed using the GIT [76] version control system. This allows keeping the full history of all code changes.

Code reviews are prevalent in the software industry, as a way to improve code quality and harmonize development practices across organizations. All LHCb GIT projects are hosted on the CERN GitLab server [77], which also provides features allowing better collaboration between the member of the teams: new features to the code are peer reviewed before being merged into the main code base (using what is known in GitLab as *merge requests*). GIT also allows several branches in parallel (e.g. for Run 2 software and Upgrade) with the possibility to merge back code from one branch to the other. The LHCb source code was first managed using the CVS version control system, then migrated to SVN and now to GIT, giving thus confidence that the industry's evolution will be followed in the future.

As the LHCb software stack depends on many components, a Software Configuration Database has been built to track all projects and their dependencies [78] using the Neo4j graph database [79]. This information is crucial to the management of the software in the short term, but is also necessary to identify the software needed to continue analysing the LHCb experiment data in the long term.

The projects are organized around the JIRA [80] task management system, as deployed by the CERN IT department. This allows the developers within the LHCb collaboration to follow the evolution of the projects and collaborate in a constructive manner.

8.2.2 Build infrastructure and continuous integration

In order to ensure the quality of the software produced by the developers, the LHCb core software team has put in place automatic builds of the software, as described in Ref. [81]. This infrastructure relies on the industry tested JENKINS [82] automation server as deployed in the CERN IT Openshift service [83]. The LHCb continuous integration system is described in figure 8.1.

Releases of the LHCb software are provided for the versions of Linux provided by the CERN IT department, and deployed on the grid (i.e. Scientific Linux CERN release 5 and 6, as well as CentOS 7 depending on the release date).

The computing resources needed to run this infrastructure are provided by the CERN IT department and are OPENSTACK [84] virtual machines [85]. This server distributes the builds to a cluster of build nodes and gathers the results which are displayed on a custom made web application [86].

The same infrastructure is used to prepare the releases of the LHCb software as shown in Fig. 8.2, with the difference that RPM packages are prepared and installed in that case as described in the next section.

Unit tests are run straight after the builds and the results are published to the LHCb Dashboard [87]. Integration tests requiring more resources can be run using the LHCb Performance and Regression testing service [56]. LHCbPR is responsible for systematically running the regression tests, collecting and comparing results of these tests so that any changes between different setups can be easily observed. The framework is based on a microservices architecture which breaks a project into loosely coupled modules communicating with each other through APIs.

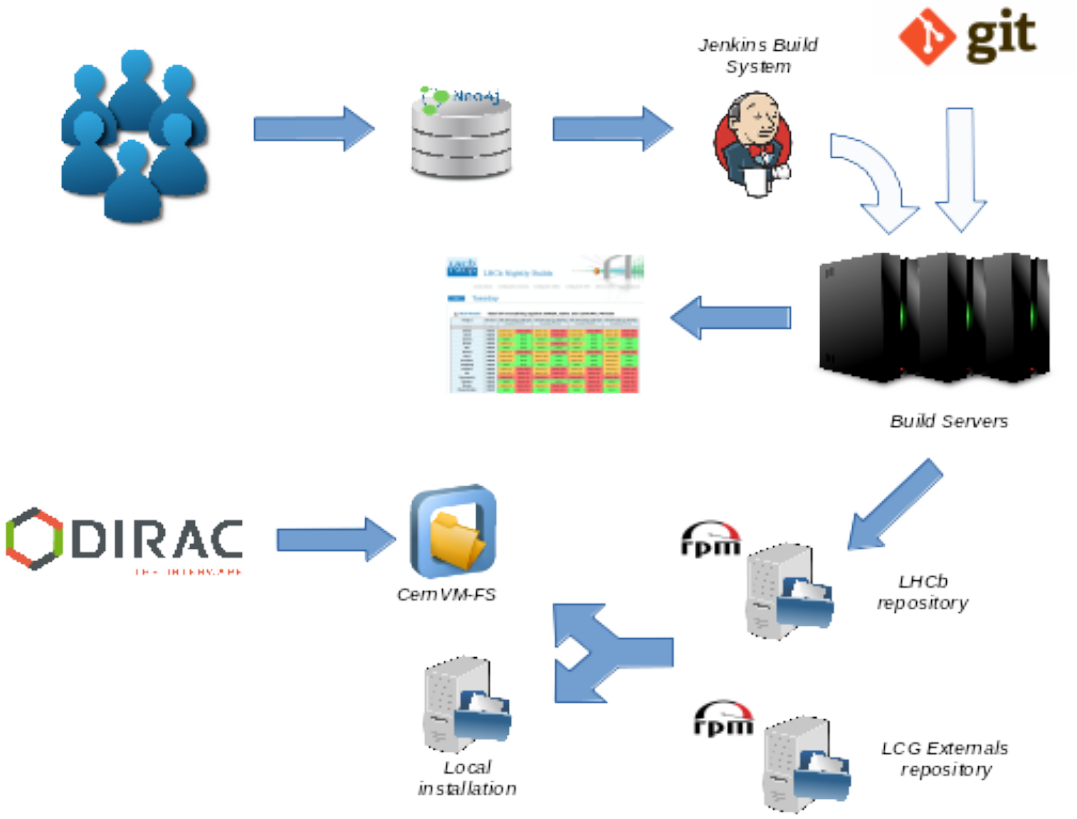


Figure 8.1: LHCb continuous integration system.

Most of the developed modules are generic which means that the proposed framework can be adopted for other experiments. Figure 8.3 presents a sequence diagram of the interaction between the test service, LHCbPR and users. The test service requests from LHCbPR information on how to run tests, then actually runs them and finally saves the results back to LHCbPR. Users have the ability to retrieve and analyse these tests results. Figure 8.4 shows an example of timing of the LHCb simulation package Gauss during the development of the application, as displayed by LHCbPR.

8.2.3 Packaging and distribution

The LHCb Software stack is composed of many different applications (see Chap. 2) that rely on many external packages (*e.g.* an installation of the analysis package relies on more than a hundred different packages). The applications are therefore packaged in the RPM Package Manager [88] format, which allows specifying dependencies. This enables private installations of the software stack, as well as full management of the installed packages.

This is however not always necessary as the LHCb software is also distributed using the CERNVM filesystem or CVMFS [89].

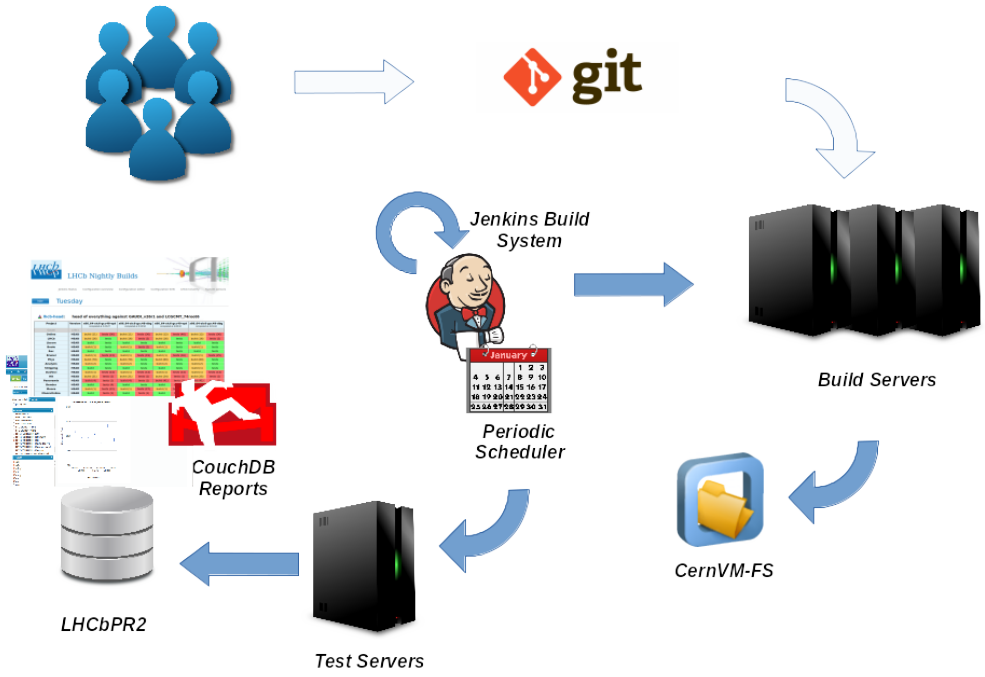


Figure 8.2: LHCb release system

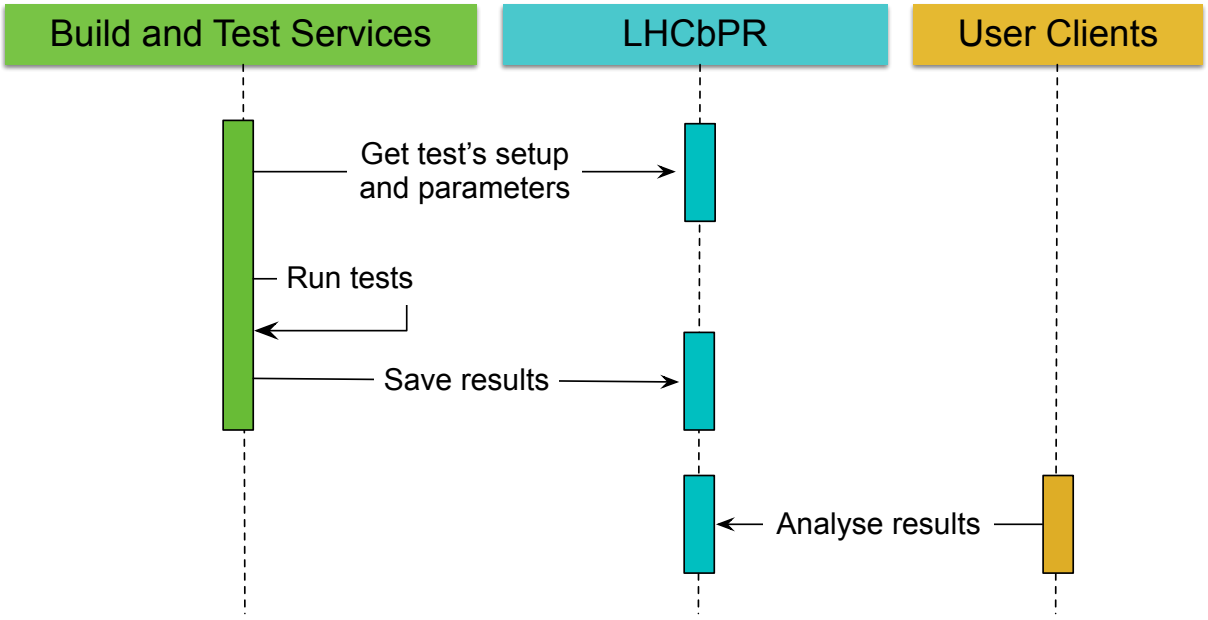


Figure 8.3: Sequence diagram of interaction between test running services, LHCbPR and users.

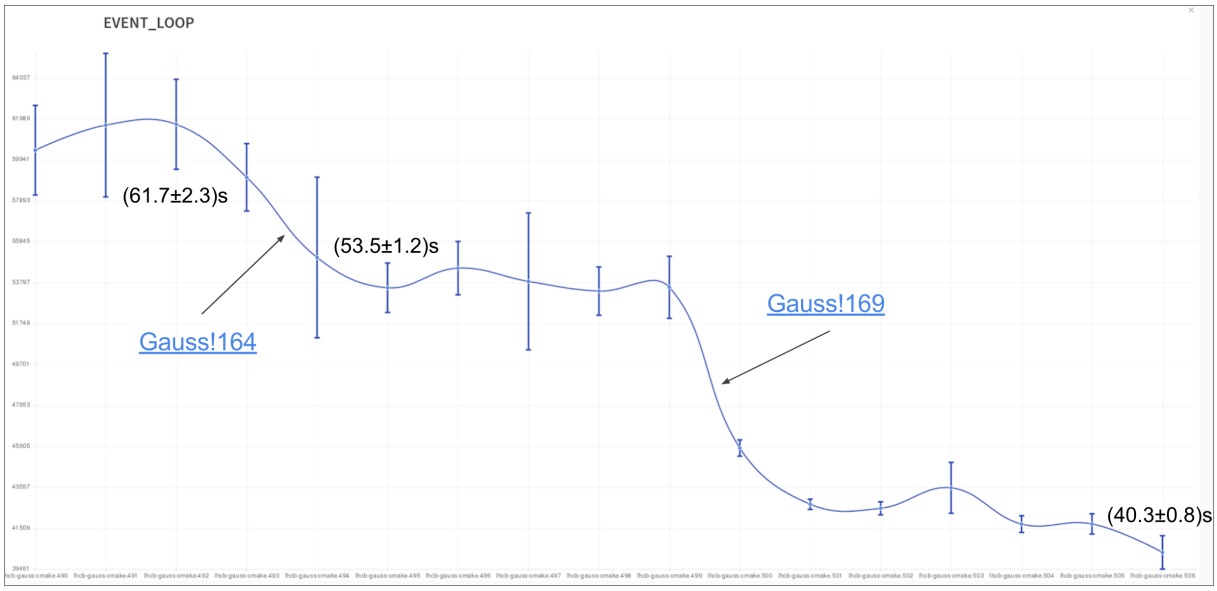


Figure 8.4: Example of timing of the LHCb Simulation application in LHCbPR.

8.2.4 Support for new and heterogeneous architectures

The current LHCb software stack is only supported on Intel architecture, and currently only on 64 bit architectures. The use of other architectures (ARM, IBM Power), as well as of GPUs (see Sec. 5), is however being investigated.

A new architecture can be handled by LHCb build and release infrastructure provided:

- the operating system used by LHCb (currently CentOS7) can be installed;
- LCGCMake packages have been ported and released;
- nodes for the architecture can be managed by the experiment's instance of JENKINS.

Once the above conditions are fulfilled, new architectures and platforms can easily be added to the LHCb build and release system.

Building, testing and deploying software for GPUs has the same pre-requisites but may impose other resource management conditions which have not yet been investigated.

8.2.5 Long term preservation

With the tools in place in the LHCb development process, it is enough to keep the version control system as well as the CVMFS repositories to make sure that the LHCb software can be kept running in the long term. Virtual machines and containers ensure that the supported platforms can still be run in the medium to long term. As it will be shown in Sec. 9, LHCb is also working on databases containing metadata about the software artefacts, their dependencies and their use in physics to make sure they can be re-built and re-run if necessary. The following systems are crucial to long term preservation:

- the source code management system to ensure a persistence of the evolutions in the code;

- the CVMFS distribution of the code (O(1TB));
- the preservation database with metadata about the projects.

8.2.6 Resources

The LHCb core software team strives to use common infrastructure provided by the CERN IT department, whenever possible, such as:

- the JIRA task management system;
- an instance of the JENKINS continuous integration system running on Openshift;
- file storage in EOS.

Building and testing the software however requires a significant cluster currently comprising:

- 45 nodes (360 Virtual cores) to build and test the software;
- 15 nodes (60 cores) to run other services needed by the system.

The LHCb upgrade does not change the amount of resources needed to build, test and package the software. However, recent developments have highlighted a need for improved integration tests (e.g. for simulation validation, or to check trigger code), that may require either an extension of the resources allocated to test and/or the possibility to test new builds on clusters such as `lxbatch`.

8.3 Collaborative tools

The design, development and maintenance of the LHCb software requires contributions from many actors. Several collaborative tools are being used in order to coordinate the efforts and increase the proficiency of software developers. Rather than developing ad-hoc solution, the general strategy is to monitor the available tools and trends and adapt them to the specific use cases of LHCb. The collaboration uses a number of these tools to implement the functions described in the following sections.

8.3.1 Communication

Within and across the different activities forms the foundation of the collaborative effort. It is important to appreciate the different levels at which communication happens, ranging from informal discussions to highly structured requests, involving only a few specialists to distributing information to the whole collaboration. A major challenge for the management of the internal communication is to ensure that relevant information is generated and reaches all affected people, while at the same time controlling the information bandwidth that needs to be digested by the individual collaborator. As any large project LHCb uses an approach that concentrates communication among the groups of relevant people. To counterbalance this compartmentalization as a general rule all technical information is made accessible to the collaboration on an on-demand basis. Various tools are being provided by CERN to manage these different levels of communication. Currently used tools include: e-groups, JIRA, GitLab issues and merge requests, MATTERMOST, TWIKI, INDICO, VIDYO. LHCb will continue relying on these tools or in whatever will be made available in the meantime. Specific LHCb communication tools include

- DIRAC *portal;* enables to submit and manage formal production requests for the central data production
- *LHCb web site;* official communication and links to various projects and services

8.3.2 Documentation

The Working Group databases record status information on various projects, in particular physics analyses. The information is organized using several tools like:

- *LHCb Q&A:* question and answers database, serves as an interactive FAQ database in the style of Stack Overflow. Currently LHCb Q&A is managed by volunteers from the collaboration. In the future it might be replaced by an installation of the original Stack Exchange software at CERN, managed by CERN IT.
- *TWiki:* the LHCb TWiki is the main tool used to create and maintain working documentation of the LHCb software (and other projects).
- *Indico:* manages meeting agendas and archives presentation slides and other materials attached to events. It is not uncommon for some of those slides to contain information that is kept available through the indico archiving function.
- *LHCbDocs:* a Subversion repository for all official LHCb documents, enables version tracking and collaborative drafting of documents. Due to the upcoming dismission of SVN, a new tool will be used. This is being discussed within the LHCb Editorial Board.
- *LHCb publication pages:* provide summaries on all LHCb publications, including links to the internal documentation, the public preprint and paper and the public plots and tables.
- *CDS;* central document database at CERN. Long term documentation is compiled into pdf documents (technical notes, design reports, analysis notes, etc) which are archived on CDS.

8.3.3 Basic training

Basic training and education of newcomers to the LHCb software are vital to keep knowledge alive and provide an effective workforce of developers. Training goes beyond providing excellent documentation and also has to take didactics into account. A very effective way to train newcomers has evolved in the form of the LHCb Starterkit [90], which is a web page containing lessons and tutorials hosted on GitHub. The Starterkit has been developed and is maintained and expanded by a community of mostly young LHCb members. Since it is made by newcomers for newcomers the resulting teaching material is very well aligned with the needs of people new to the LHCb software. There are two one-week long training courses offered per year, which are based on this tutorial. The Starterkit maintainers are largely recruited from the participants of those training events. Due to the excellent quality and the collaborative platform afforded by the GitHub host, the community has proven to be self-sustaining. Beyond providing essential training in the LHCb software and development tools, the Starterkit promotes software literacy and best practices.

8.3.4 Advanced training

The challenges posed by the code and algorithmic optimizations have been addressed by organizing software *hackathons* approximately every two months. During one week, participants attend to tutorials and lectures on basic (*e.g.* C++ course, hardware architectures) and advanced topics (*e.g.* memory management, vectorization techniques) and have the opportunity of improving the performance of their code by interacting with software and computing experts and working with them. This enables to make rapid progress on software development in specific domains.

8.3.5 Collaborative software development

A well organized collaborative software development requires specialized tools (version control system, issue tracker, etc) to streamline the workflow as discussed in detail in Sec. 8.2.1.

Chapter 9

Data and analysis preservation

The data collected at the LHC and the analyses performed on that data have a special significance owed to the scale of the project. A repetition of the experiment and a reproduction of the obtained results carries such a huge cost that, in order to adhere to the scientific principles of reproducibility, dedicated practices are justified. On the other hand, the fact that data collection and analysis is performed by a large collaboration composed of several institutes and their respective teams of researchers shifts the focus of reproducibility with respect to practices adopted in other sciences. The situation in HEP requires custom answers to the challenges raised by the principle of reproducibility.

One particular problem is the unusually high complexity of data manipulation required to obtain a typical physics result. Even the space permitted in the most verbose of journal papers does not permit to describe the procedures on a level of detail that would be required to reproduce a result. Indeed it is acknowledged that the reproducing a result is a challenge even with all the traditional internal documentation, as long as the computer code used to perform a data analysis is not available. Therefore reproducibility is tightly linked to preservation of the data as well as the analysis code itself and constitutes a complex problem in knowledge management.

The large number of collaborators, the long time scales of the experiment and the complexity of the data manipulations necessitate to look at the problem on different scales. On the lowest level any analysis preservation scheme already facilitates knowledge transfer within the original team of proponents. It benefits the review process when internal reviewers have to scrutinize the analysis. Other analysis teams can profit from work previously done most effectively if they can reuse those developments with minimal effort. Finally a knowledge transfer to future generations of physicists requires access to the tools and the unique data set used in current research.

CERN aims at providing a common infrastructure for analysis preservation to all experiments. The project is named CERN Analysis Preservation Portal (CAP) [91], it is open source and publicly shared [92].

The CAP web portal collects meta data about analyses in an online form. The meta data include information about data provenance, production setup and software, analysis code, documentation, publications and workflows. CAP offers a web-based database form to submit this information but the system is also being prepared to ingest data from various sources, such as GitLab repositories, the LHCb working group and publication databases and the stripping pages.

CAP aims to provide the ability to rerun analyses. A dedicated project called Reusable Analyses or REANA [93] has been started to develop this service. It is a joint effort of

RECAST [94], DASPOS [95] and CAP with a goal to run physics analysis on the cloud. The project uses industry standard technologies such as Docker [96] containers, that are running on a Kubernetes [97] cluster. CERN experiments are required to use this interface to submit their analyses for central preservation.

In order to make best use of these central services provided by CERN LHCb has developed a technical and conceptual strategy to prepare for the reproducibility challenge [98]. This strategy is based on two principles and identifies practices in four domains. It has been inspired by the recent debate about best practices for reproducibility [99] and adapts solutions proposed by the wider research community to the special situation in HEP. These principles are summarised in the following.

9.1 Preservation of pre-processed data

Partial analysis preservation focuses on the preservation of pre-processed data. This principle is driven by constraints imposed by the large size of the collected data sets and the limited amount of computing resources available to the collaboration and the wider community. In LHCb the initial data processing is done centrally using highly developed and well cross-checked software. The central data processing up to and including the stripping is the heavy lifting in terms of handling large data volumes. Even after this step the amount of data, which is available to the individual analysts is enormous ($\sim 1\,\text{TB}$ for the Run 1 legacy stripping). To deal with this problem the analysis preservation strategy will only attempt to preserve the heavily preselected data sets actually used for each individual analysis in form of the ntuples (or filtered micro DSTs). Earlier data stages will be archived but not made available in the sense of active preservation outlined in principle 2.

9.2 Active analysis preservation

Active analysis preservation goes beyond the simple archiving of data, code and documentation usually associated with preservation efforts. In order to get a return on the invested efforts on all of the levels of knowledge transfer outlined above, analysis preservation needs to accomplish analysis reuse. Active preservation not only requires that the code is preserved but that it can be executed successfully by a third party. This requirement becomes more important with increasing analysis complexity and is similar to the requirement that documentation of a complex software package comes with working tutorials and examples. This second principle leads to the questions on how to automate analysis workflows and how to preserve the runtime environment necessary to execute the analysis software. The answer to these questions is given by recent progress in software development, in particular the introduction of continuous integration as best practice (cf Sec. 8.2.1). Active analysis preservation is an adoption of these already existing practices and tools to the problem of analysis development and preservation (see also Ref. [100]).

9.3 The four domains of analysis preservation

Best practices and tools are being developed in the following four domains.

- *Data preservation:* the preprocessed data, which serves as input to an analysis needs to be available under credentials managed by the collaboration and be archived on a reliable

storage service. Storage will be organized through the physics working groups, who will provide dedicated locations for each analysis (and possibly shared data) on the CERN central EOS storage. Long time archiving will be achieved through the CAP infrastructure. The latter will also provide mechanisms to prevent accidental removal of the data. CERN e-groups provide convenient means of controlling the access to the data.

- *Code repository:* all code of preserved analyses is kept in a git repository managed through the CERN GitLab interface. Again access credentials will be managed through CERN e-groups. Additionally, CAP will enable to capture snapshots of the repository for secure archiving.

- *Analysis automation:* In order to make an analysis reusable the workflow should be automated as much as possible. Automatic analysis pipelines are a widely discussed pattern and several software tools are available (see e.g. Refs. [100–103]). The details of how and to which extend an analysis can be automatize will be left to the proponents. The collaboration will provide tutorials and training to enable more analysis teams to adopt a fully automated pipeline.

- *Runtime environment:* The analysis code usually depends on a software stack and external libraries to function properly. Following the lead by the CAP project LHCb will adopt Linux container technology to capture the runtime environments for each individual analysis. Basic Docker images will be provided for analysts to base their docker containers on. In addition templates for the analysis GitLab repositories will be provided thus enabling analysis teams to make use of the special cloud tenants provided by CERN for the specific purpose of building docker containers.

Analysis preservation requires an evolution of collaborative practices. The collaboration aims at providing the necessary tools and infrastructure, fully based on available technology, to support and facilitate this process.

Chapter 10

Project organization, planning and risk register

This chapter presents an overview of the organization of the Computing Project towards the Run 3 upgrade and its interaction with related bodies within the experiment and outside. It also presents an high-level planning of the activities, with milestones, as well as a risk register.

The hardware costs for data processing are discussed in the Trigger and Online TDR [2] for online resources, and in the process of the resource scrutiny by the CERN Computing Resources Scrutiny Group (CRSG) and Resource Review Board (RRB) for offline resources. An overview of the offline computing and storage organisation is provided in a separate document describing the computing model [10]

10.1 Project organization

The Computing Project currently consists of five subprojects covering the different areas of software and computing within LHCb (see Fig. 10.1). In contrast to the work towards the LHC startup, LHCb is currently operating a detector and processing data, activities that still require a significant part of the person-power of the Computing Project. The operations and development for the current detector are covered by the subprojects for Applications Software, Distributed Computing, Data Handling and Services&Operations. The upgrade activities are concentrated in the fifth subproject which will be absorbed by the others at the startup of Run 3.

The Run 3 upgrade subproject covers the areas of development in core software, distributed computing and collaborative tools. Core software development encompasses the upgrade of the framework software into a task parallel framework, the upgrade of the event model, detector description, conditions data and the engineering work needed for the new simulation framework. The area of distributed computing entails the further evolution of the LHCbDirac middleware for data handling, the development of centralized working group productions, train processing and the Ganga project for individual analysis on distributed computing resources. Collaborative tools also incorporate data preservation, analysis preservation and open data aspects of the experiment. The upgrade activities are also supported by the other computing subprojects, such as Services&Operations and Data Handling.

While it is expected that computing activities represent a common area for the whole collaboration, and it is expected that people from across the collaboration contribute to this

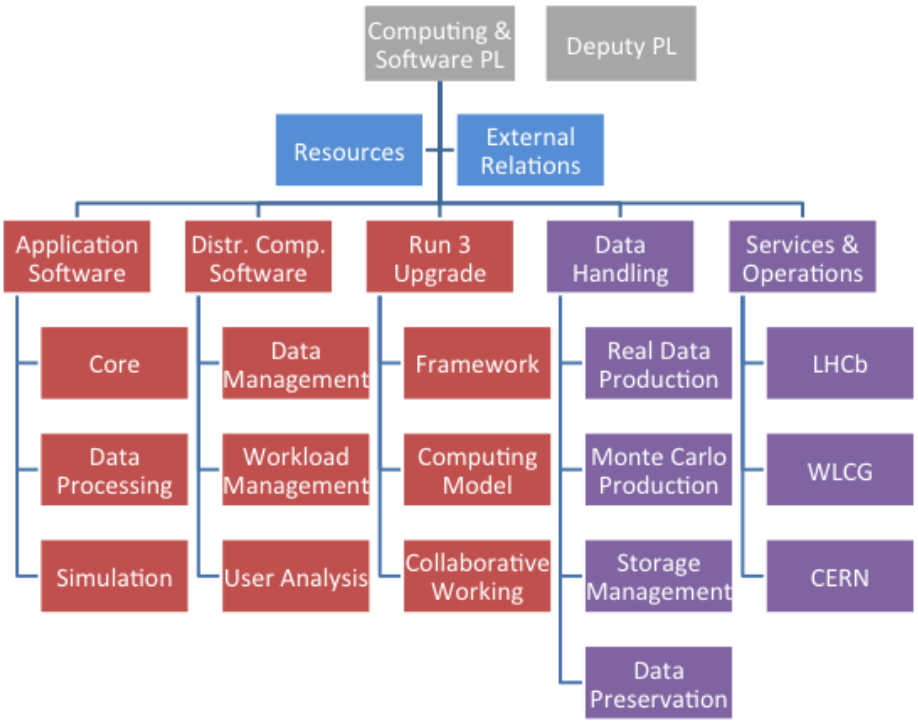


Figure 10.1: Organization of the Computing Project.

area, the Computing Project has also the possibility to conclude Core Computing Agreements with LHCb institutes. These agreements are signed by the members of the National Computing Board (NCB, see below) and keep record of primary commitments of institutes to support core computing and software. The contributions can be given in any area of the Computing Project in form of tasks and roles, whose descriptions are provided in the agreement. Both tasks and roles are limited in time and scope. A contribution that has reached the end of its time limit can be finished, renewed or re-described. New contributions can also be included in retrospect.

The Computing project leader and the NCB chair are responsible for the setup and follow-up of the institutional contributions. New revisions of the Core Computing Agreements will be released indicatively on a yearly basis in a dedicated NCB session.

Any differences arising during the execution of these agreements will be first mediated by the project leader and, if not resolved, submitted to the Spokesperson of LHCb who will propose solutions to the best interest of the Collaboration. The agreements are not legally binding, but the institutes recognize that the success of the LHCb experiment depends on the functioning of its Computing Project.

A distribution of person-power in the different areas of the Computing Project in 2018, as resulting from the current version of the Core Computing Agreements, is given in Fig. 10.2. The

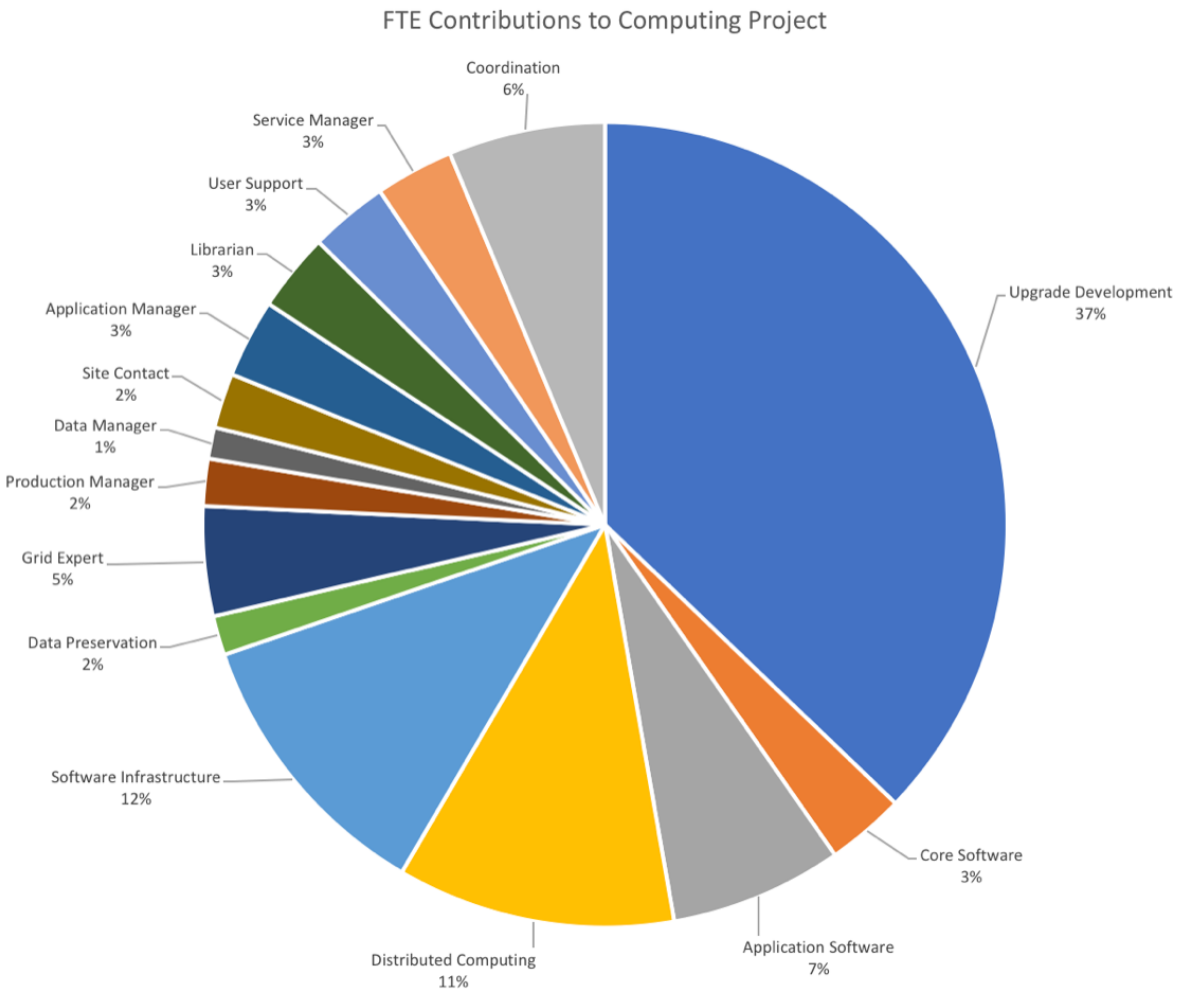


Figure 10.2: Distribution of person-power in the Computing Project in 2018.

total efforts correspond to 31.1 FTEs, 37% of which are devoted to upgrade developments purely related to core computing.

Given that there are many common development areas with other projects such as High Level Trigger, detector sub-systems, and physics performance working groups such as Tracking and Alignment, Particle Identification and Calorimeter Objects, it is not straightforward, and it is not either necessary, to strictly assign person-power to each of them. The Computing Project and these other projects are part of the *Upgrade Software Planning Group* (USPG), which by itself is the body that coordinates the computing activities together with the other projects in a wider scope. The USPG oversees the upgrade software activities, ensures coordination among all the projects and working groups in order to optimize efforts and resources and facilitates effective connections between the core software and the sub-system reconstruction software developers. The Computing Project participates to the USPG with the offline data processing coordinator and the upgrade computing coordinator.

In addition to the above structure the National Computing Board serves as a body to

disseminate information from the Computing Project towards the collaborating institutes for what concerns the hardware and person-power resource needs for the software and computing development and operational tasks. The NCB representatives are responsible for collating and disseminating all relevant information for discussion within the countries they represent. They sign the Core Computing Agreement (see above). The NCB serves as a forum to discuss common approaches and problems within home institutes and countries. It advises the LHCb Collaboration Board, LHCb management and Computing Project on external computing issues, including computing resource requirements and person-power for the core computing and software.

10.2 Work packages and institutional contributions

The tasks of the upgrade work are the following.

- Core Software
  - Framework
  - Event Model
  - Non-event data
  - Alternative architectures
  - Applications
- Distributed Computing
  - Distributed Computing Infrastructure (DIRAC)
  - Centralized User Analysis
  - Individual User Analysis (GANGA)
- Simulation
  - Simulation performance optimisation
  - Interaction with core software
- External Dependencies and Infrastructure
  - External software dependencies
  - Infrastructure
- Collaborative Working
  - Collaborative Tools
  - Data and analysis preservation
  - Open data access

The following institutes are currently contributing or have signed up to contribute to the software and computing upgrade activities.

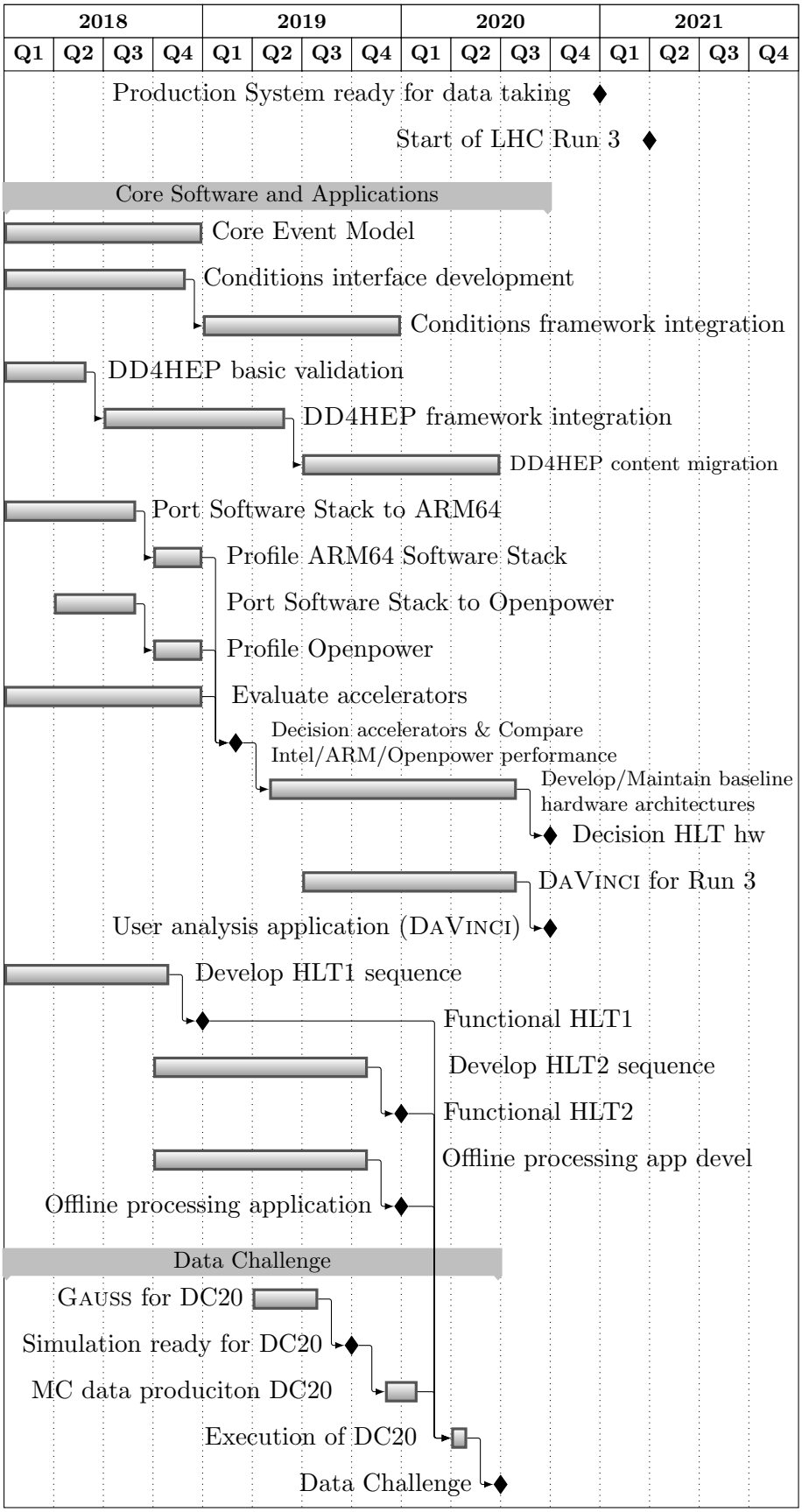
- University of Chinese Academy of Sciences, Beijing, China

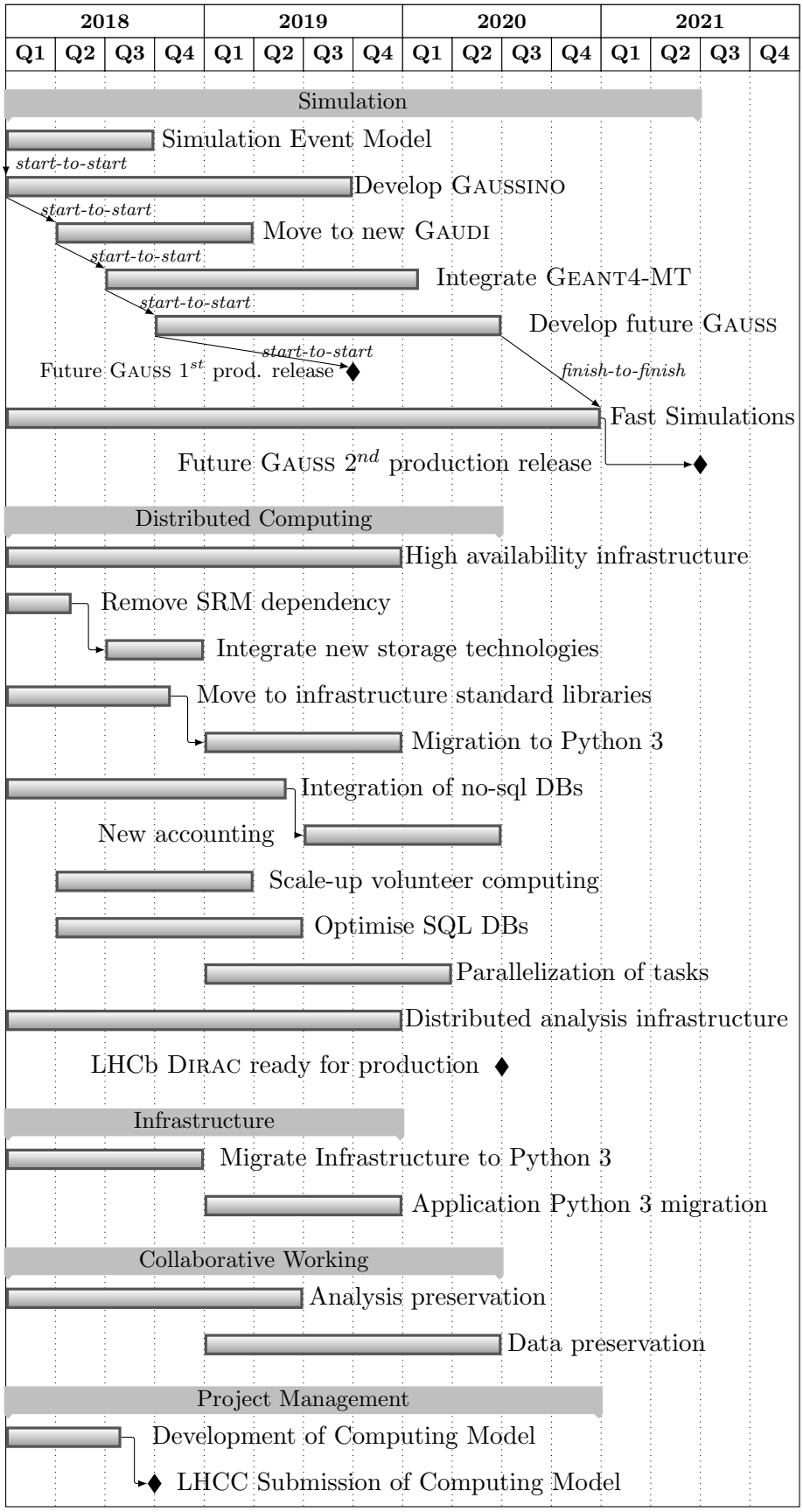
- Center for High Energy Physics, Tsinghua University, Beijing, China
- Aix Marseille Univ, CNRS/IN2P3, CPPM, Marseille, France
- LAL, Univ. Paris-Sud, CNRS/IN2P3, Université Paris-Saclay, Orsay, France
- LPNHE, Université Pierre et Marie Curie, Université Paris Diderot, CNRS/IN2P3, Paris, France
- Sezione INFN di Cagliari, Cagliari, Italy
- Sezione INFN di Ferrara, Ferrara, Italy
- Laboratori Nazionali dell'INFN di Frascati, Frascati, Italy
- Sezione INFN di Pisa, Pisa, Italy
- Fakultaet Physik, Technische Universitaet Dortmund, Dortmund, Germany
- AGH - University of Science and Technology, Faculty of Physics and Applied Computer Science, Krakow, Poland
- Henryk Niewodniczanski Institute of Nuclear Physics Polish Academy of Sciences, Krakow, Poland
- National Center for Nuclear Research (NCBJ), Warsaw, Poland
- Horia Hulubei National Institute of Physics and Nuclear Engineering, Bucharest-Magurele, Romania
- Yandex School of Data Analysis, Moscow, Russia
- National Research Tomsk Polytechnic University, Tomsk, Russia
- European Organization for Nuclear Research (CERN), Geneva, Switzerland
- Nikhef National Institute for Subatomic Physics and VU University Amsterdam, Amsterdam, The Netherlands
- University of Birmingham, Birmingham, United Kingdom
- Imperial College London, London, United Kingdom
- School of Physics and Astronomy, University of Manchester, Manchester, United Kingdom
- Department of Physics, University of Warwick, Coventry, United Kingdom
- University of Cincinnati, Cincinnati, OH, United States
- Massachusetts Institute of Technology, Cambridge, MA, United States

10.3 Milestones and planning

The following Gantt chart summarizes the main acitivites, their time development, and the relationship between them. A series of milestones is also given.

It is foreseen to have a complete "vertical slice" test of the upgrade data flow (starting from HLT1 all the way through to the offline selections) in the first quarter of 2020. The purpose of the test is to validate the complete functionality of the HLT and offline data flows, including details of calibration, streaming, data exchange etc., as well as physics validation of the reconstruction and selection algorithms. In order to have meaningful numbers of events coming through all stages of the selections, a minimum of 1 billion minimum bias events will need to be simulated in upgrade conditions, and be available by the end of 2019. The detector response simulation (Boole) of the upgrade detector will thus have to be ready by mid-2019.


2018
2019
2020
2021
Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
Production System ready for data taking
Start of LHC Run 3
Core Software and Applications
Core Event Model
Conditions interface development
Conditions framework integration
DD4HEP basic validation
DD4HEP framework integration
DD4HEP content migration
Port Software Stack to ARM64
Profile ARM64 Software Stack
Port Software Stack to Openpower
Profile Openpower
Evaluate accelerators
Decision accelerators & Compare
Intel/ARM/Openpower performance
Develop/Maintain baseline
hardware architectures
Decision HLT hw
DaVinci for Run 3
User analysis application (DaVinci)
Develop HLT1 sequence
Functional HLT1
Develop HLT2 sequence
Functional HLT2
Offline processing app devel
Offline processing application
Data Challenge
Gauss for DC20
Simulation ready for DC20
MC data produciton DC20
Execution of DC20
Data Challenge


2018
2019
2020
2021
Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
Simulation
Simulation Event Model
start-to-start
Develop GAUSSINO
start-to-start
Move to new GAUDI
start-to-start
Integrate GEANT4-MT
start-to-start
Develop future GAUSS
start-to-start
Future GAUSS 1st prod. release
finish-to-finish
Fast Simulations
Future GAUSS 2nd production release
Distributed Computing
High availability infrastructure
Remove SRM dependency
Integrate new storage technologies
Move to infrastructure standard libraries
Migration to Python 3
Integration of no-sql DBs
New accounting
Scale-up volunteer computing
Optimise SQL DBs
Parallelization of tasks
Distributed analysis infrastructure
LHCb DIRAC ready for production
Infrastructure
Migrate Infrastructure to Python 3
Application Python 3 migration
Collaborative Working
Analysis preservation
Data preservation
Project Management
Development of Computing Model
LHCC Submission of Computing Model

10.4 Copyright and license

The centrally managed and distributed software stack of LHCb currently does not contain an explicit copyright statement nor a license. LHCb wants to fix these two points in order to comply with applicable laws and other regulations. The goal will be to organize the software in this respect in a way that it can also be managed in the future from a legal point of view in a simple way. This process will include interactions with all collaborating institutes of the collaboration as well as legal advisers.

10.5 Risk register

The risks associated to the work described in this document have been evaluated and are described in this section. The process of risk analysis is following well established procedures described *e.g.* in Ref. [104] of planning risk management, identify risks, perform qualitative and quantitative risk analysis, planning risk responses, and monitor and control risks.

The remainder of this section contains a risk register of the of the work described in this report, following the guidelines above. Updates on the register will be communicated to the LHCC on a regular basis during LHCC meetings and in case of request by the LHCC during upgrade reviews.

From the data processing chain point of view, the main risk is represented by not matching the performance of the software trigger in event processing at 30 MHz inelastic event rate. This risk affects different projects in LHCb, *i.e.* computing, trigger, online, tracking&alignment, and the sub-detector projects. Despite the progress observed since the start of the development work, the final goal of fitting the required data processing performance within the budget of the online farm has not been achieved yet. The impact of not achieving the goal is high as it has consequences on the physics performance of the experiment. Mitigation measures include engineering choices, such as offloading parts of the processing to resources outside the event filter farm, or physics choices reducing the efficiency of different physics channels.

The part related to core software engineering on the framework includes risks in the architectural changes foreseen for the modernization of the Gaudi framework in the areas of detector description and conditions data. Both areas currently contain outdated and unmaintained software packages. The work in those areas is progressing and the risk is considered to be small. In case the modernization of those packages is not successful, the previous versions need to be maintained, taking into account the fact that they are not compatible as such with a multi-threaded software environment.

The availability of offline resources for storing and distributing the recorded data on distributed computing resources affects the physics performance of the experiment. Given the increase in luminosity of the experiment data taking, both computing and storage needs will further increase beyond what has been requested during Run 2 data taking. The impact of this risk is a delay in the physics program of the experiment. Mitigation measures include: optimization of the simulation performance for the computing resources and possible parking of data for later analysis. A description of a computing model with further details and the needed resources in the different areas will be provided to the LHCC in the third quarter of 2018.

The experiment distributed computing infrastructure for processing and offline data storage needs to scale with the increased data demands. The impact of not being able to deal with the data in this area is high as it will impact any further data processing from offline handling

of the data to user analysis work. To deal with this risk, in addition to the work plan laid out in this document, the individual components need to be monitored with respect to their scaling, and engineering work needs to be done in case of issues. The architectural design of the LHCb/DIRAC distributed computing infrastructure allows for individual components to be easily replaced. Therefore, the risk of the whole infrastructure to be incompatible with the Run 3 demands is considered to be small.

References

[1] LHCb collaboration, *Framework TDR for the LHCb Upgrade: Technical Design Report*, CERN-LHCC-2012-007. LHCb-TDR-012.

[2] LHCb collaboration, *LHCb Trigger and Online Technical Design Report*, CERN-LHCC-2014-016. LHCb-TDR-016.

[3] M. Frank *et al.*, *Deferred High Level Trigger in LHCb: A Boost to CPU Resource Utilization*, J. Phys. Conf. Ser. **513** (2014) 012006.

[4] G. Dujany and B. Storaci, *Real-time alignment and calibration of the LHCb Detector in Run II*, J. Phys. Conf. Ser. **664** (2015) 082010.

[5] R. Aaij *et al.*, *Tesla: an application for real-time data analysis in High Energy Physics*, Comput. Phys. Commun. **208** (2016) 35, `arXiv:1604.05596`.

[6] R. Aaij *et al.*, *Upgrade trigger: Biannual performance update*, LHCb-PUB-2017-005.

[7] Geant4 collaboration, S. Agostinelli *et al.*, *Geant4: A simulation toolkit*, Nucl. Instrum. Meth. **A506** (2003) 250.

[8] Geant4 collaboration, J. Allison *et al.*, *Geant4 developments and applications*, IEEE Trans. Nucl. Sci. **53** (2006) 270.

[9] Geant4 collaboration, J. Allison *et al.*, *Recent developments in Geant4*, Nucl. Instrum. Meth. **A835** (2016) 186.

[10] LHCb collaboration, *Computing Model of the Upgrade LHCb experiment*, CERN-LHCC-2018-014. LHCb-TDR-018.

[11] LHCb collaboration, *LHCb computing: Technical Design Report*, CERN-LHCC-2005-019. LHCb-TDR-011.

[12] I. Bird *et al.*, *Update of the Computing Models of the WLCG and the LHC Experiments*, Tech. Rep. CERN-LHCC-2014-014. LCG-TDR-002, Apr, 2014. `http://cds.cern.ch/record/1695401`.

[13] R. Brun and F. Rademakers, *ROOT: An object oriented data analysis framework*, Nucl. Instrum. Meth. **A389** (1997) 81.

[14] J. T. Mościcki *et al.*, *Ganga: A tool for computational-task management and easy access to Grid resources*, Computer Physics Communications **180** (2009) 2303 .

[15] ALICE collaboration, M. Zimmermann, *The ALICE analysis train system*, J. Phys. Conf. Ser. **608** (2015) 012019, arXiv:1502.06381.

[16] P. Mato, *GAUDI-Architecture design document*, Tech. Rep. LHCb-98-064, CERN, Geneva, Nov, 1998. https://cds.cern.ch/record/691746.

[17] G. Barrand *et al.*, *Gaudi – a software architecture and framework for building HEP data processing applications*, Comput. Phys. Commun. **140** (2001) 45.

[18] *Task-based Programming*, https://software.intel.com/en-us/node/506100. [Online; accessed 7-May-2018].

[19] *The Concurrent Framework Project (CF4Hep)*, http://concurrency.web.cern.ch/GaudiHive, 2012. [Online; accessed 7-May-2018].

[20] I. Shapoval *et al.*, *Graph-based decision making for task scheduling in concurrent Gaudi*, in *IEEE Nuclear Science Symposium & Medical Imagine Conference Record*, November, 2015. doi: 10.1109/NSSMIC.2015.7581843.

[21] I. Shapoval, *Adaptive Scheduling Applied to Non-Deterministic Networks of Heterogeneous Tasks for Peak Throughput in Concurrent Gaudi*, PhD thesis, Mar, 2016, http://cds.cern.ch/record/2149420, presented 18 Mar 2016.

[22] K. Rupp, *40 years of microprocessor trend data*, https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data. [Online; accessed 7-May-2018].

[23] *DDR4 set to account for largest share of DRAM market by architecture*, http://electroiq.com/blog/2017/04/ddr4-set-to-account-for-largest-share-of-dram-market-by-architecture/. [Online; accessed 7-May-2018].

[24] *The adversarial relationship of the DRAM user and producer continues*, http://electroiq.com/blog/2017/09/the-adversarial-relationship-of-the-dram-user-and-producer-continues/. [Online; accessed 7-May-2018].

[25] *Simd vector classes for c++*, https://github.com/VcDevel/Vc. [Online; accessed 7-May-2018].

[26] *Experimental range library for C++11/14/17*, https://github.com/ericniebler/range-v3, 2014. [Online; accessed 7-May-2018].

[27] *SoAContainer*, https://gitlab.cern.ch/LHCbOpt/SOAContainer, 2017. [Online: accessed 7-May-2018].

[28] *PODIO, or plain-old-data I/O*, https://github.com/hegner/podio, 2016. [Online; accessed 7-May-2018].

[29] S. Roiser, *Event Data Modelling for the LHCb Experiment at CERN*, PhD thesis, 2003, https://cds.cern.ch/record/692288, presented on 30 Sep 2003.

[30] A. Valassi *et al.*, *CORAL and COOL during the LHC long shutdown*, J. Phys. Conf. Ser. **513** (2014) 042045.

[31] R. Sipos *et al.* , *Functional tests of a prototype for the CMS-ATLAS common non-event data handling framework*, J. Phys. Conf. Ser. **898** (2017) 042047.

[32] *DD4hep (Detector Description for High Energy Physics)*, `https://github.com/AIDASoft/DD4hep`, 2015. [Online; accessed 7-May-2018].

[33] S. Ponce, P. Mato Vila, A. Valassi, and I. Belyaev, *Detector description framework in LHCb*, eConf **C0303241** (2003) THJT007, `arXiv:physics/0306089`.

[34] *USolids/VecGeom*, `http://aidasoft.web.cern.ch/USolids`, 2015. [Online; accessed 7-May-2018].

[35] *Guide for ECS FSM design in LHCb sub-detectors, LHCb EDMS note 655828*, `https://edms.cern.ch/ui/file/655828/3/LHCb_ECS_FSM_Guidelines.pdf`.

[36] F. Bonifazi *et al.*, *The Message Logger for the LHCb on-line farm*, Tech. Rep. LHCb-2005-050. CERN-LHCb-2005-050, CERN, Geneva, Aug, 2005. `http://cds.cern.ch/record/877498`.

[37] M Frank *et al.* , *The LHCb high level trigger infrastructure*, J. Phys. Conf. Ser. **119** (2008) 022023.

[38] M. De Cian *et al.*, *Status of HLT1 sequence and path towards 30 MHz*, Tech. Rep. LHCb-PUB-2018-003. CERN-LHCb-PUB-2018-003, CERN, Geneva, Mar, 2018. `http://cds.cern.ch/record/2309972`.

[39] *Fact Sheet: Collaboration of Oak Ridge, Argonne, and Livermore (CORAL)*, [Online: accessed 7-May-2018], `https://energy.gov/downloads/fact-sheet-collaboration-oak-ridge-argonne-and-livermore-coral/`.

[40] *Inside Amazon's Cloud Computing Infrastructure*, `https://www.linkedin.com/pulse/inside-amazons-cloud-computing-infrastructure-masoud-ostad`.

[41] *IEEE Standard for Floating-Point Arithmetic*, IEEE Std 754-2008 (2008) 1.

[42] D. H. Cámpora Pérez, O. Awile, and C. Potterat, *A high-throughput Kalman filter for modern SIMD architectures*, in *Euro-Par 2017: Parallel Processing Workshops*, p. 378, Springer International Publishing, 2018. doi: 10.1007/978-3-319-75178-8_31.

[43] T. Sjöstrand, S. Mrenna, and P. Skands, *PYTHIA 6.4 physics and manual*, JHEP **05** (2006) 026, `arXiv:hep-ph/0603175`.

[44] T. Sjöstrand, S. Mrenna, and P. Skands, *A brief introduction to PYTHIA 8.1*, Comput. Phys. Commun. **178** (2008) 852, `arXiv:0710.3820`.

[45] M. Dobbs *et al.*, *HepMC 2, a C++ Event Record for Monte Carlo Generators - User Manual*, `http://lcgapp.cern.ch/project/simu/HepMC/HepMC20110/HepMC2_user_manual.pdf`.

[46] I. Belyaev *et al.*, *Handling of the generation of primary events in Gauss, the LHCb simulation framework*, J. Phys. Conf. Ser. **331** (2011) 032047.

[47] I. Belyaev *et al.*, *Integration of GEANT4 with the GAUDI framework*, Proceedings of CHEP2001, Beijing, China (2001), `http://cds.cern.ch/record/1745074`.

[48] S. Porteboeuf, T. Pierog, and K. Werner, *Producing Hard Processes Regarding the Complete Event: The EPOS Event Generator*, in *Proceedings, 45th Rencontres de Moriond on QCD and High Energy Interactions: La Thuile, Italy, March 13-20, 2010*, pp. 135–140, Gioi Publishers, Gioi Publishers, 2010. `arXiv:1006.2967`.

[49] D. J. Lange, *The EvtGen particle decay simulation package*, Nucl. Instrum. Meth. **A462** (2001) 152.

[50] J. Alwall *et al.*, *A Standard format for Les Houches event files*, Comput. Phys. Commun. **176** (2007) 300, `arXiv:hep-ph/0609017`.

[51] G. Amadio *et al.*, *The GeantV project: preparing the future of simulation*, J. Phys. Conf. Ser. **664** (2015) 072006.

[52] J. de Favereau *et al.*, *DELPHES 3, A modular framework for fast simulation of a generic collider experiment*, JHEP **02** (2014) 057, `arXiv:1307.6346`.

[53] D. Müller, *Measurements of production cross-sections and mixing of charm mesons at LHCb*, PhD thesis, Nov, 2017, `https://cds.cern.ch/record/2297069`, presented 23 Oct 2017.

[54] J. Schaarschmidt, *The new ATLAS Fast Calorimeter Simulation*, J. Phys. Conf. Ser. **898** (2017) 042006.

[55] I. J. Goodfellow *et al.*, *Generative Adversarial Networks*, `arXiv:1406.2661`.

[56] A. Mazurov, B. Couturier, D. Popov, and N. Farley, *Microservices for systematic profiling and monitoring of the refactoring process at the LHCb experiment*, J. Phys. Conf. Ser. **898** (2017) 072037.

[57] F. Stagni *et al.*, *DIRAC in Large Particle Physics Experiments*, J. Phys. Conf. Ser. **898** (2017), no. 9 092020.

[58] Z. Mathe, A. C. Ramo, N. Lazovsky, and F. Stagni, *The DIRAC Web Portal 2.0*, J. Phys. Conf. Ser. **664** (2015) 062039.

[59] A. C. Ramo, R. G. Diaz, and A. Tsaregorodtsev, *DIRAC RESTful API*, J. Phys. Conf. Ser. **396** (2012) 052019.

[60] V. M. Muñoz *et al.*, *The Integration of CloudStack and OCCI/OpenNebula with DIRAC*, J. Phys. Conf. Ser. **396** (2012) 032075.

[61] F. Stagni *et al.*, *DIRAC universal pilots*, J. Phys. Conf. Ser. **898** (2017) 092024.

[62] F. Stagni *et al.*, *LHCbDirac: distributed computing in LHCb*, J. Phys. Conf. Ser. **396** (2012) 032104.

[63] Z. Mathe and P. Charpentier, *Optimising query execution time in LHCb Bookkeeping System using partition pruning and Partition-Wise joins*, J. Phys. Conf. Ser. **513** (2014) 042032.

[64] F. Stagni, P. Charpentier, and the LHCb Collaboration, *The LHCb DIRAC-based production and data management operations systems*, J. Phys. Conf. Ser. **368** (2012) 012010.

[65] F. Stagni, A. Tsaregorodtsev, A. McNab, and C. Luzzi, *Pilots 2.0: DIRAC pilots for all the skies*, J. Phys. Conf. Ser. **664** (2015) 062061.

[66] P. Andreetto *et al.*, *CREAM Computing Element: a status update*, J. Phys. Conf. Ser. **396** (2012) 032003.

[67] A. Filipčič, D. Cameron, and J. K. Nilsen, *Dynamic Resource Allocation with the arcControlTower*, J. Phys. Conf. Ser. **664** (2015) 062015.

[68] B. Bockelman *et al.*, *Commissioning the HTCondor-CE for the Open Science Grid*, J. Phys. Conf. Ser. **664** (2015) 062003.

[69] A. McNab, *The vacuum platform*, J. Phys. Conf. Ser. **898** (2017) 052028.

[70] N. Høimyr *et al.*, *BOINC service for volunteer cloud computing*, J. Phys. Conf. Ser. **396** (2012) 032057.

[71] C. Haen and B. Couturier, *LHCbDIRAC as Apache Mesos microservices*, J. Phys. Conf. Ser. **898** (2017) 092016.

[72] R. Currie *et al.*, *Expanding the user base beyond HEP for the Ganga distributed analysis user interface*, J. Phys. Conf. Ser. **898** (2017) 052032.

[73] *LCG Release Area*, `https://ep-dep-sft.web.cern.ch/document/lcg-releases`. [Online: accessed 7-May 2018].

[74] *LCGCmake*, `https://gitlab.cern.ch/sft/lcgcmake`. [Online: accessed 7-May 2018].

[75] *HEP_Oslibs*, `https://twiki.cern.ch/twiki/bin/view/LCG/CentOS7DependencyRPM`. [Online: accessed 7-May 2018].

[76] *git distributed version control system*, `https://git-scm.com/`. [Online: accessed 7-May 2018].

[77] *CERN GitLab instance*, `http://gitlab.cern.ch`. [Online: accessed 7-May 2018].

[78] A. Trisovic, B. Couturier, V. Gibson, and C. Jones, *Recording the lhcb data and software dependencies*, J. Phys. Conf. Ser. **898** (2017) 102010.

[79] *neo4j graph database*, `https://neo4j.com/`. [Online; accessed 7-May-2018].

[80] *Jira Software*, `https://www.atlassian.com/software/jira`. [Online; accessed 7-May-2018].

[81] M. Clemencic and B. Couturier, *LHCb Build and Deployment Infrastructure for run 2*, J. Phys. Conf. Ser. **664** (2015) 062008.

[82] *Jenkins*, https://jenkins.io/. [Online; accessed 7-May-2018].

[83] http://openshift.cern.ch/. [Online; accessed 7-May-2018].

[84] https://www.openstack.org/. [Online; accessed 7-May-2018].

[85] https://openstack.cern.ch/project/. [Online; accessed 7-May-2018].

[86] M. Clemencic, B. Couturier, and S. Kyriazi, *Improvements to the User Interface for LHCb's Software continuous integration system.*, J. Phys. Conf. Ser. **664** (2015) 062025.

[87] *LHCb Nightly builds dashboard*, https://lhcb-nightlies.cern.ch/nightly/summary/. [Online; accessed 7-May-2018].

[88] *RPM Package Manager*, http://rpm.org/. [Online; accessed 7-May-2018].

[89] *CERN VM File System (CVM-FS)*, https://cernvm.cern.ch/portal/filesystem. [Online; accessed 7-May-2018].

[90] L. Bel, *The LHCb Starterkit*, PoS **ICHEP2016** (2017) 334.

[91] J. Cowton *et al.*, *Open data and data analysis preservation services for lhc experiments*, J. Phys. Conf. Ser. **664** (2015) 032030.

[92] *CAP source code*, https://github.com/cernanalysispreservation/analysispreservation.cern.ch. [Online; accessed 7-May-2018].

[93] *REANA - Reusable Analyses*, http://reana.readthedocs.io/en/latest/index.html. [Online; accessed 7-May-2018].

[94] K. Cranmer and I. Yavin, *Recast — extending the impact of existing analyses*, Journal of High Energy Physics **2011** (2011) 38.

[95] M. D. Hildreth, *Data and software preservation for open science (daspos)*, in *2014 IEEE 30th International Conference on Data Engineering Workshops*, pp. 142–146, March, 2014. doi: 10.1109/ICDEW.2014.6818318.

[96] *Docker*, http://docker.io. [Online; accessed 7-May-2018].

[97] *Kubernetes*, http://kubernetes.io. [Online; accessed 7-May-2018].

[98] S. Neubert *et al.*, *LHCb Analysis Preservation Roadmap*, LHCb-INT-2017-021, https://cds.cern.ch/record/2280615.

[99] V. Stodden and S. Miguez, *Best practices for computational science: Software infrastructure and environments for reproducible and extensible research*, SSRN (2013) .

[100] B. Beaulieu-Jones and C. Green, *Reproducibility of computational workflows is automated using continuous analysis*, Nature Biotechnology **35** (2017) 342–346.

[101] I. Jimenez *et al.*, *Popper: Making Reproducible Systems Performance Evaluation Practical*, UCSC-SOE (2016), no. 16-10, https://www.soe.ucsc.edu/research/technical-reports/UCSC-SOE-16-10.

[102] J. Köster and S. Rahmann, *Snakemake – a scalable bioinformatics workflow engine*, Bioinformatics **28** (2012) 2520.

[103] P. Amstutz *et al.*, *Common Workflow Language, v1.0*, , `https://figshare.com/articles/Common_Workflow_Language_draft_3/3115156`.

[104] *A Guide To The Project Management Body Of Knowledge (PMBOK Guides)*, Project Management Institute, 2004.