Parallel Computing Studies for the Alignment of the ATLAS Silicon Tracker

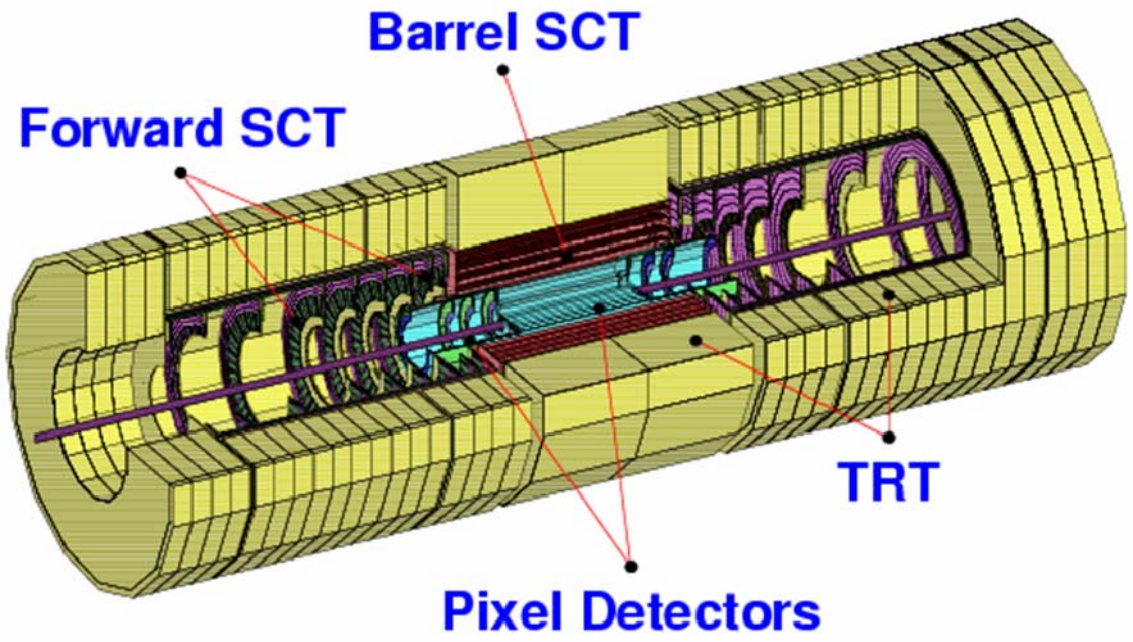
Müge Karagöz Ünel
(the University of Oxford)
K. Bernardet, P. Brückman, A. Hicheur
the Atlas Collaboration

Computing in HEP,
T.I.F.R., Mumbai
13-17 February, 2006

- Motivation: Alignment of Si-tracker
- Tests with random matrices
- Case-studies using ATLAS MC data

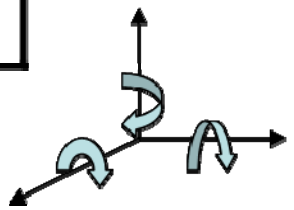
Temple in Tamil Nadu, www.sacredcities.com

The Fact: Atlas Silicon System has ~ 35k DoF



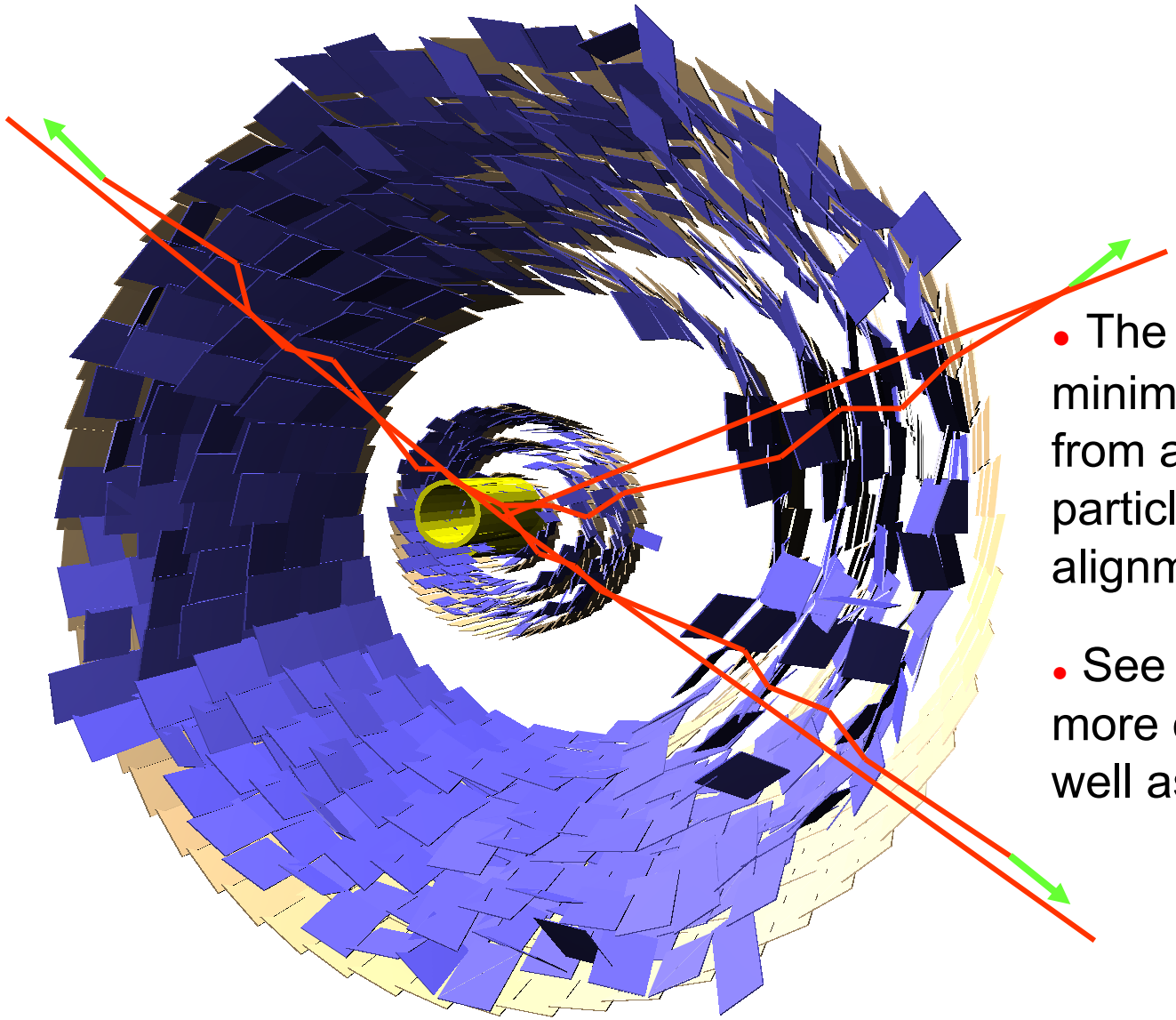
	Barrel		Forward	
Detector	PIX	SCT	PIX	SCT
# of layers/disks	3	4	2x3	2x9
# of modules	1456	2112	2x144	2x988
sub Total	3568		2264	
Total	5832			

3 translations
& 3 rotations
of each module



In total we have to deal with 34,992 DoF's!

The Idea: Global χ^2 Method for Alignment



- The method consists of minimizing a giant χ^2 resulting from a simultaneous fit of all particle trajectories and alignment parameters.
- See Adlene Hicheur's talk for more on Global χ^2 algorithm as well as ATLAS alignment issues.

The Problem: Limitations

- Global χ^2 formalism requires solving large number of linear equations and handling large sized-matrices in its full scope, due to the large number of DoF in the Atlas tracker system.

- Using 32-bit single-CPU platforms is a limiting factor in performance (ATLAS-Marseille):

  - **Size:** * A matrix of N=35k needs 9.8GB disk space and at least the same amount of virtual space for computations. * A 32-bit program can only address 4GB of virtual memory. * By default, Atlas software infrastructure allows only 2GB of memory per job.

  - **Precision:** Conventional 32-bit libraries are not fully adequate for full size solution computational accuracy (35k-DoF). The matrices of alignment problems can have large condition numbers which may compete with the machine precision.

  - **Execution time:** Single-CPU machines can take a good number of hours to solve large-size problems, even if they can!

The Solution: //-processing with 64-bit Architecture

Solution: 64-bit architecture with parallel processing

and distributed-memory libraries for matrix algebra!

- **Size:**
  - e.g., distributed on 16 processors, required memory for 35k-size matrix is only 0.6GB/CPU.
  - Allowance of up to 16GB virtual memory.
- **Precision:** Hope to increase the numerical precision, making use of the enlarged data type sizes and extended-precision libraries.
- **Execution times:** Parallel processing allows faster execution times distributing the CPU load.

The Platform: Beowulf cluster at RAL (SCARF)

- SCARF is a 64-bit AMD Opteron cluster at Rutherford Lab, shared by various groups within the CCLRC users.

- Its hardware matches (and exceeds) the suggested requirements for Atlas alignment algorithms.

	Asgard (ETH Zurich)	Suggested	SCARF (RAL)
Architecture	32-bit	64-bit	64bit
# nodes	dual, 240	Large N	dual, 128
CPU	PIII 500/600 MHz	AMD Opteron 2 GHz	AMD Opteron 248 2.2. GHz
RAM	1 G	4G	4G (122 nodes), 8G (16 nodes)
Persistent	-	40 GB IDE	2x 120 GB IDE
Network	-	1x Gigabit	2x Gigabit

The Platform: Available Software

- SCARF executes x86 code natively.
- It runs Red Hat Enterprise (ES) Linux 3.0.
- Equipped with PGI and GNU compilers for C, C++ and Fortran,
- LSF (v6.1) for queuing system and MPI (v1.2.6..14a) for message-passing,
- AMD Core Math Library (ACML) (v2.5.0) with ScaLAPACK (v1.7) and BLACS (v1.1).
- We used AMD ScaLAPACK libraries for 64-bit compilation.

The Choices: Available Infrastructure

- Available compiler options on SCARF: GNU (GCC 3.2.3-53) vs PGI (5.2).
- Pros&cons:
  - GNU is publicly available, PGI is not.
  - GNU long double size is 16B on 64-bit architecture. PGI does not support quad. precision (l.d. size is 8B).
- We test time and precision performance, by calculating A=B*3-1, B=1.3.
  - Time performances are slightly better for optimized (O3) GNU. 3 orders of magnitude improvement in precision from double to l.d. with GNU. (Note: AMD Opteron holds 80-bit f.p. registers & l.d. f.p. is a 80-bit extended type.)
- We chose to continue to use GNU compiler.

The Choices: Possible Software and Libraries

- A variety of matrix algebra and solver software exists. Some examples:
  - ScaLAPACK and PLAPACK (based on LAPACK)
  - NAG parallel libraries (contains also ScaLAPACK)
- Pros & cons for ScaLAPACK:
  - AMD-ScaLAPACK is optimized for high performance on AMD Opteron.
  - Previous tests showed that ScaLAPACK eigenvalue solver (upcoming slides) performs faster than that of PLAPACK's (HPCx group).
  - ScaLAPACK (non-optimized) is a public domain software and can be downloaded from the web. NAG libraries require licence. However, this means they come with guaranteed tech. support!
  - No general quad. precision library exists for NAG or SCALAPCK… ScaLAPACK has been working on implementation for a new release. NAG already has in place iterative refinement procedures – not real extended precision, however, slows down the solution. Preconditioning the system should help in any case.
- We used ScaLAPACK on SCARF for our initial proof-of-principle studies.

Performance using Randomly-Generated Matrices

- We tested direct matrix inversion, with *pdgesvx* of ScaLAPACK, an expert driver routine to solve linear equations.
- AMD ScaLAPACK for distributed-memory computing for matrix operations; MPI for intra-processor communication, were used.
- The test matrices are symmetric, generated randomly, kept in binary format, double precision. Sizes scale as ~ N*N*8B.
- Processing time and accuracy as a function of Size(A) and N(CPU) were measured.
- A scan of Size(A) and N(CPU) for various block sizes of the matrix distribution was also obtained.

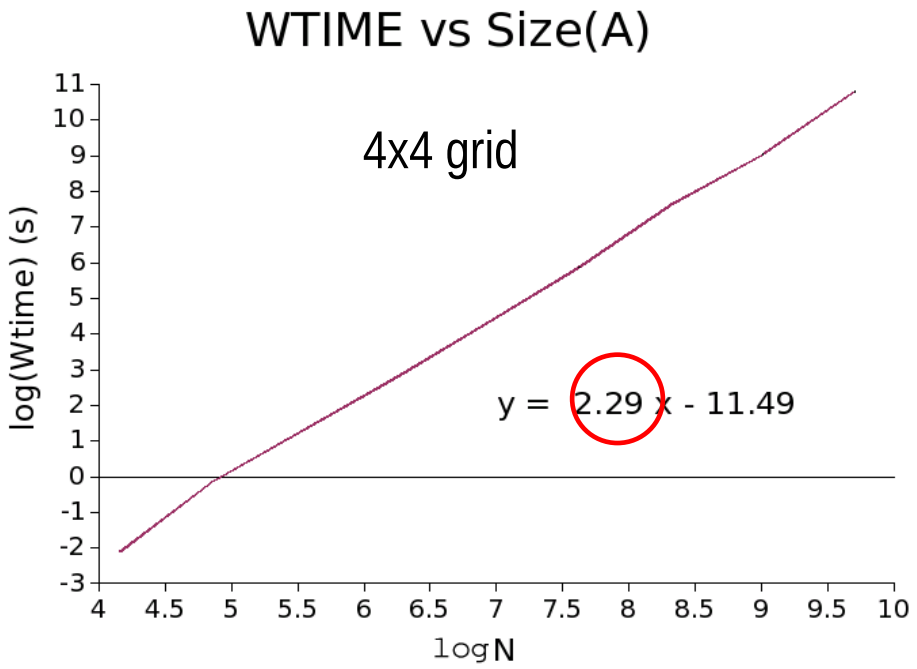
Size(A)	16384
N(CPU)	16 (4x4)
Size(Block)	64x64
Reciprocal Cond(A)	0.000000024834271
Machine ε	0.0000000000000011102
||A*A-1-I||	0.000000047526382
WallTime (for routine)	49005.65s

Performance in Processing Time:

- We measured the wall and CPU times for the ScaLAPACK routine.
- We observed minor dependence of performance on the processor grid block size. However, we found that 32 CPUs with 64MB is optimum for the range we'd like to cover.
- For an 8000 matrix, WallTime shows ~ $N(CPU)^{-0.2}$ dependence.
- 64-bit AMD showed and improvement of a factor of 2.5 in WallTime when compared with a 32-bit AMD @ 2.4GHz using the same parameters.
- We 2D-scanned the time performance in NCPU and S(grid).
- Large CPU cases not easy to test:

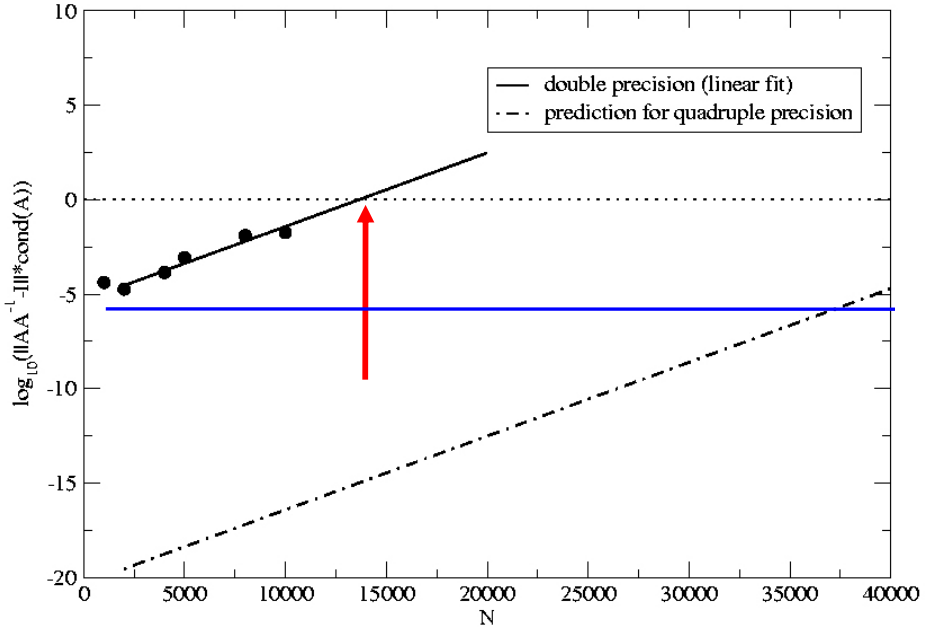
queue-waiting long – it's a shared machine!

- The results for the scan is linked here



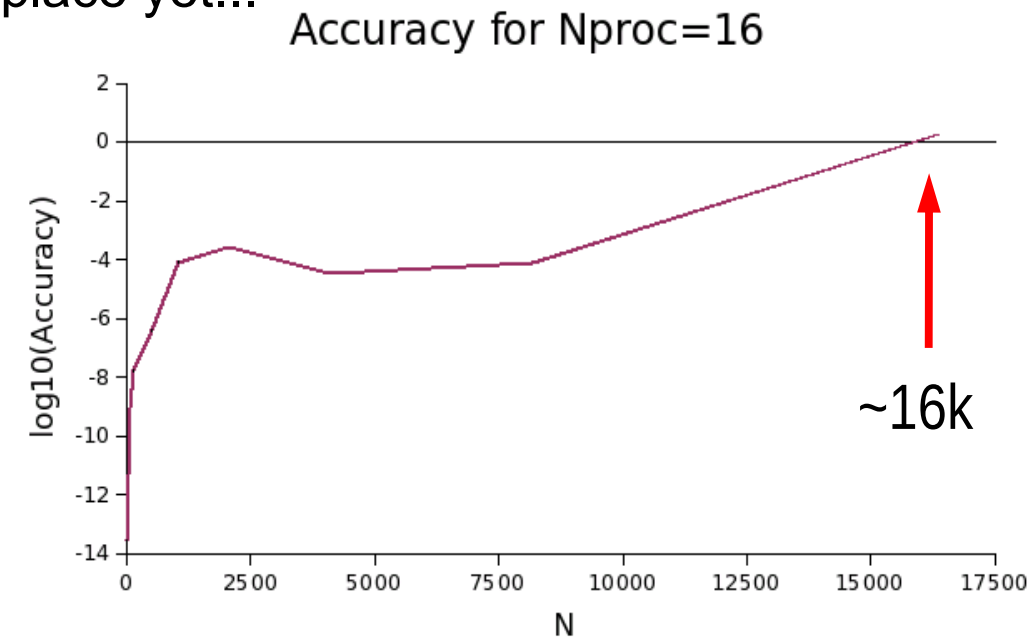
Earlier Results with a 32-bit //-cluster:

- Random matrix Inversion results using system specified earlier (Asgard)
- As a function of size of random matrices: N^3 dependence of CPU time.
- Accuracy loss at around sizes of N=14k.
- No scan of parameters as a function of grid block size was performed.
- Quadrupole precision was tested on MAPLE for N=200 (processing times enormously long) and results used to extrapolate to large sample sizes.



Performance in Precision

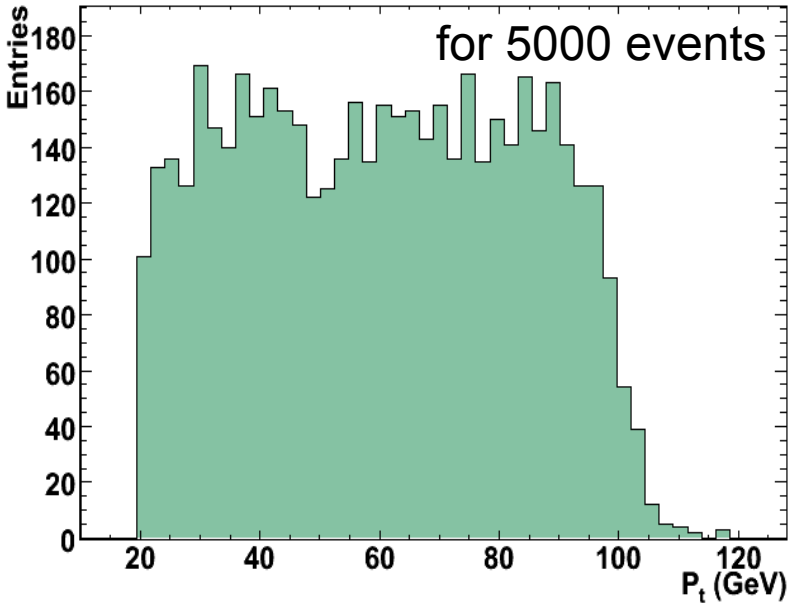
- The number of accurate decimal digits is related to the exponent of condition number of matrix: cond(A).
- Number of digits preserved in the final solution can be calculated as acc= log[(AA^T -I) x cond(A)].
- The precision loss in the final result is improved over earlier results (~N=16k), however not substantially, as we are not able to use quadrupole precision.
- Note: ScaLAPACK has plans on extended precision (iterative refinement using XBLAS), but not in place yet...



Atlas MC Data – The Setup and the Sample

- A single μ sample was simulated and reconstructed in ATLAS inner detector. 825k events exist in the sample: ~1/2 being μ^+.
- Generation, simulation and reconstruction were performed using a very recent release of the ATLAS software framework (ATHENA, v11.0.3).
- Production was carried out on the Oxford Particle Physics Linux cluster, running SLC3, using PBS. One full chain per event took ~5-10 s, integrated processing time of ~1000 hrs .
- Track transverse momenta distribution is flat: ***20 < pT< 100 GeV***.
- η-range is chosen with a flat distribution: **[-2, 2]**, φ is full and uniform.
- Full setup consist of barrel Pix+SCT region, which corresponds to **3568** modules, and **21408** DoF.

Note: As track statistics are not abundant, we will discuss solution performances rather than try to demonstrate the ultimate achievable precision in the solution properties.



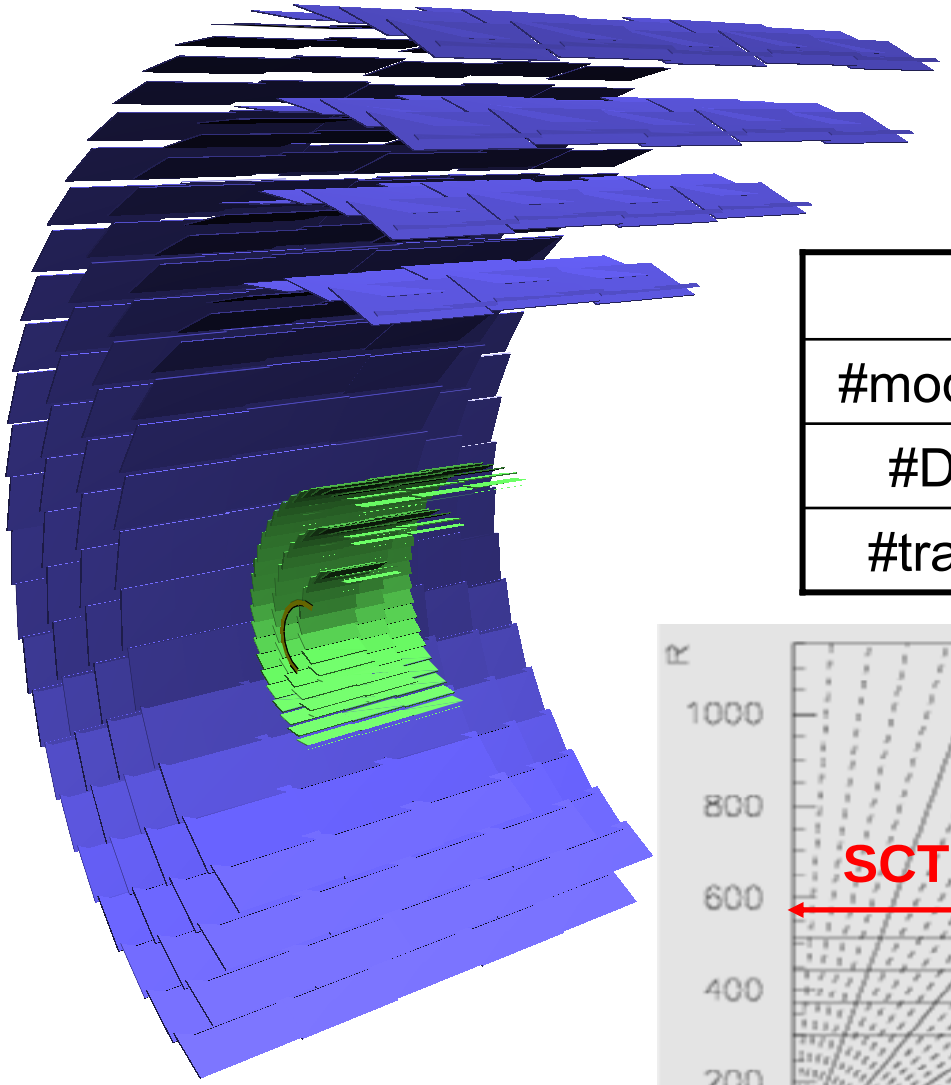
The Software Setup for Solution

- After track processing, the global χ^2 algorithm generates a binary matrix to be solved for the eigenvalues and eigenvectors of the problem.
- For this study, we used a standalone version of the algorithm, ported it also to SCARF, and solved the matrix on the cluster.
- The solution was then used to extract the alignment corrections.
- We compared two platforms for the solver: SCARF vs Intel Pentium machines with 2GB RAM, 2GB swap, and 2.8/2 GHz CPU.

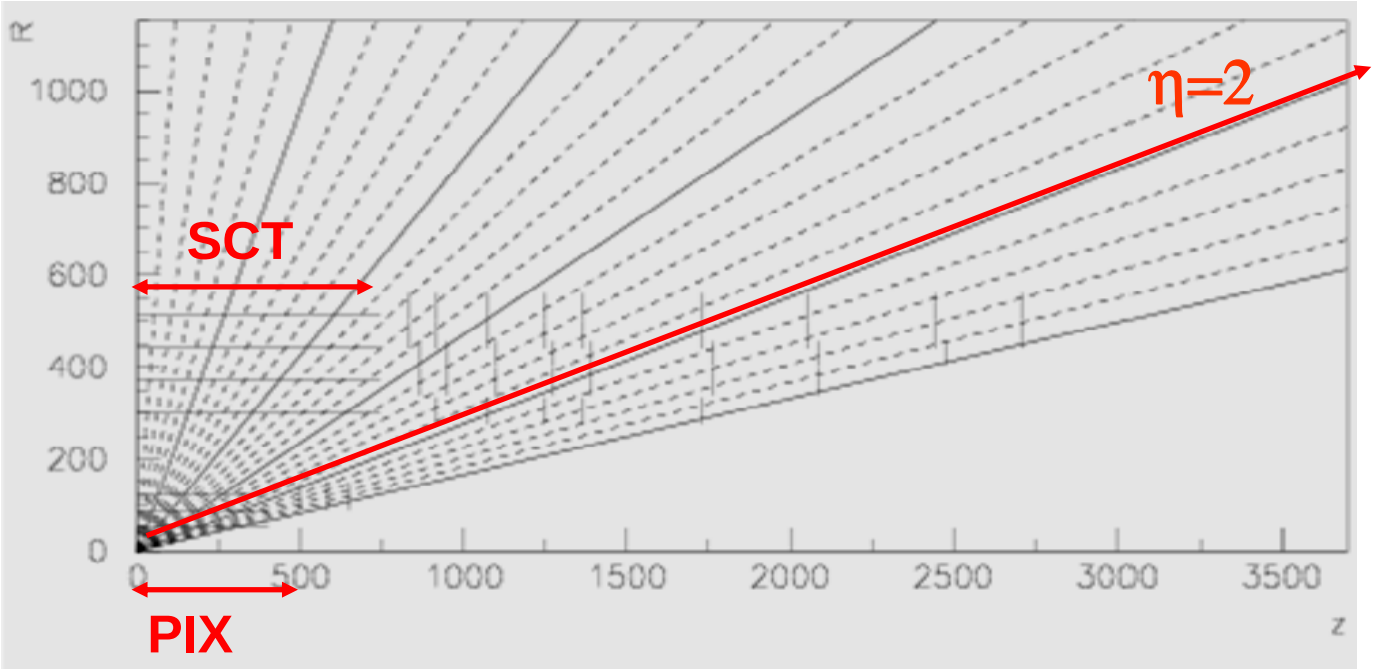
	Intel PentiumIV (32-bit)	//-cluster AMD Opteron (64-bit)
Library	LAPACK	ScaLAPACK
Routine	dspev	pdsyevd
Compiler	g++ (GCC 3.2.3-49)	gcc (GCC 3.2.3-53)
Linux version	2.4.21-32.0.1.ELsmp	2.4.21-37.ELsmp

- ***pdsyevd***: a simple Divide-and-Conquer routine (i.e., recursive reduction of an instance of the problem into one or more smaller instances.)

Solving Various Size Problems for Barrel Tracker:

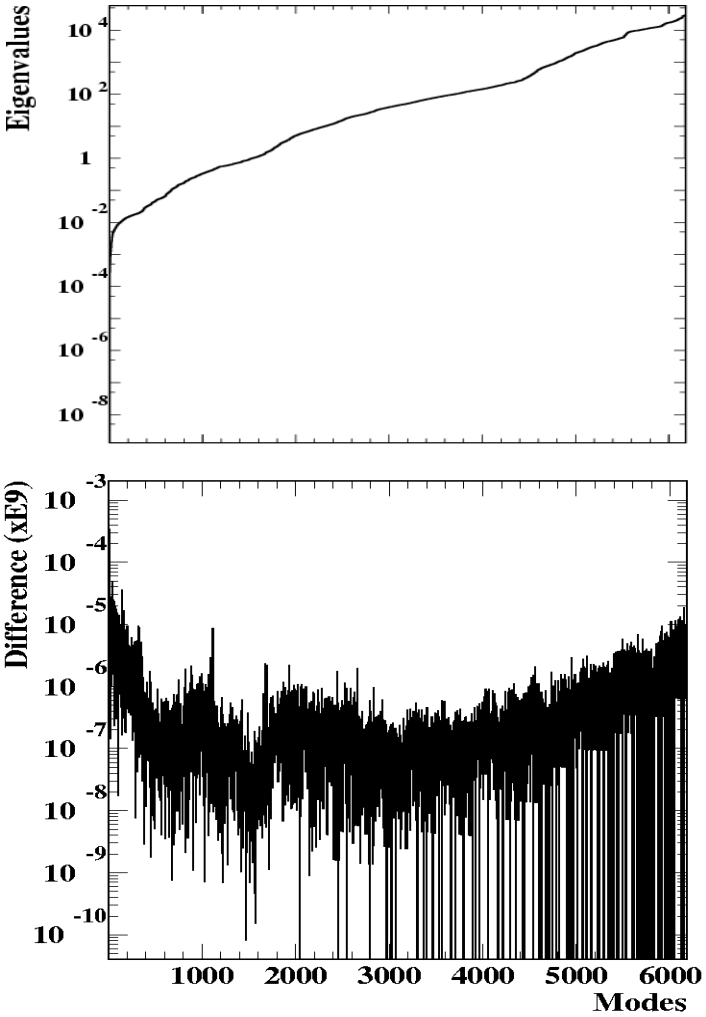


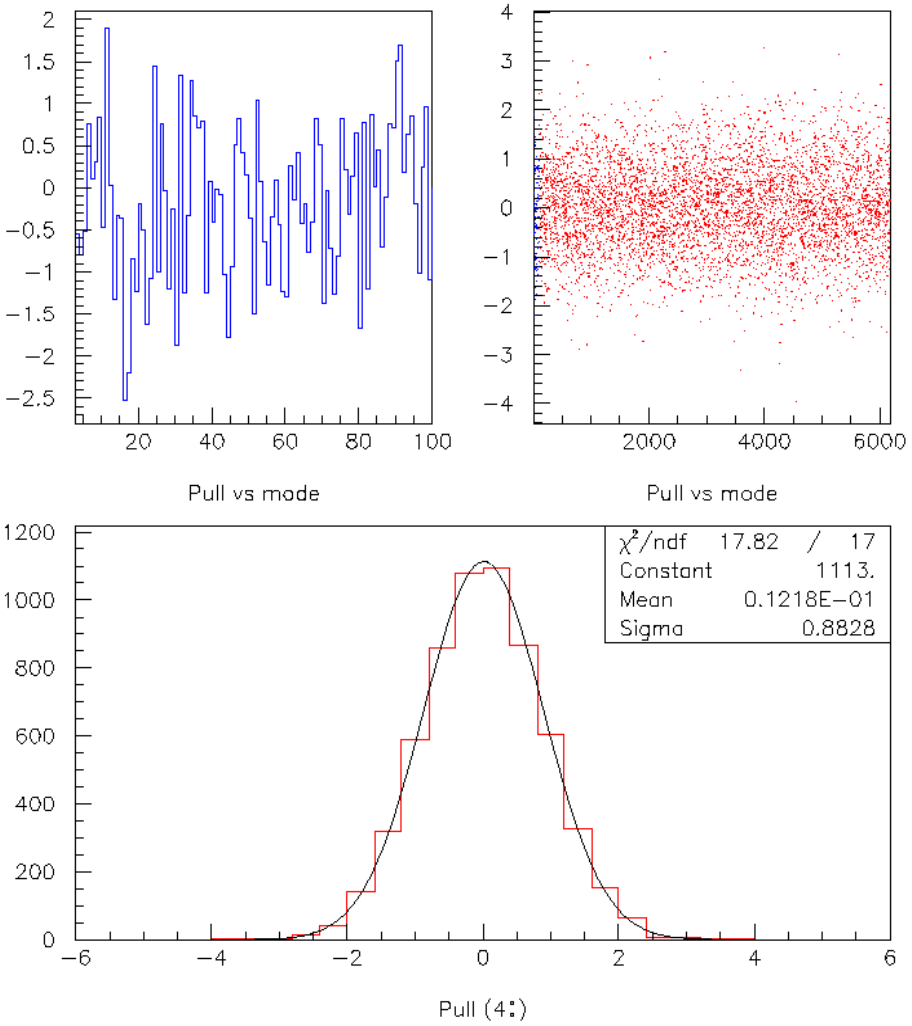
	cone	SCT	PIX+SCT
#modules	1030	2112	3568
#DoF	6180	12672	21408
#tracks	~250k	~550k	~800k



Start with the “cone” Size Problem (6k-DoF)

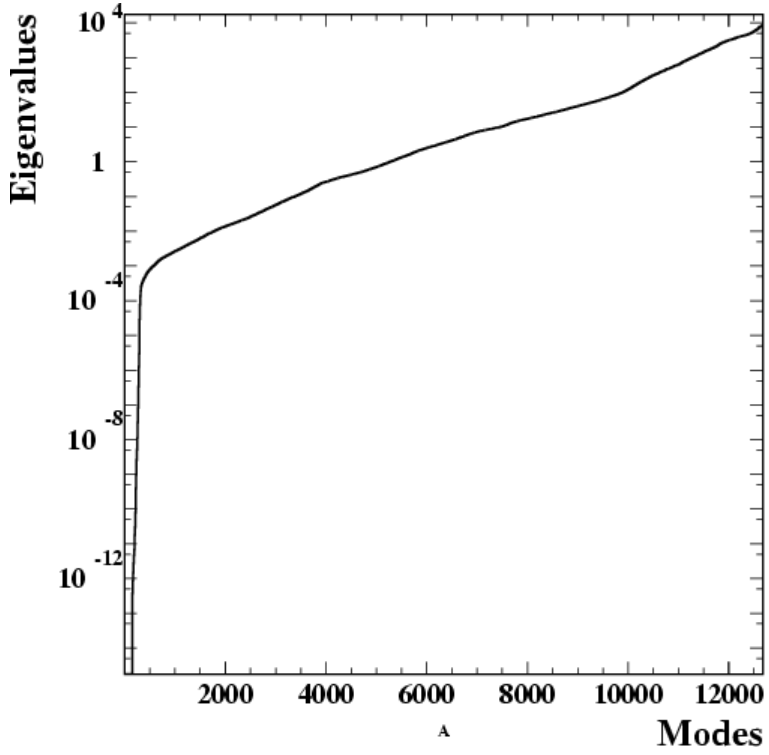
- **Cone** Configuration: Align the limited geometry by using muon tracks in restricted η (~1030 PIX+SCT modules).
- This configuration is the only one studied extensively earlier: small problem size.
- Agreement between Single-CPU and //-CPU is very good.

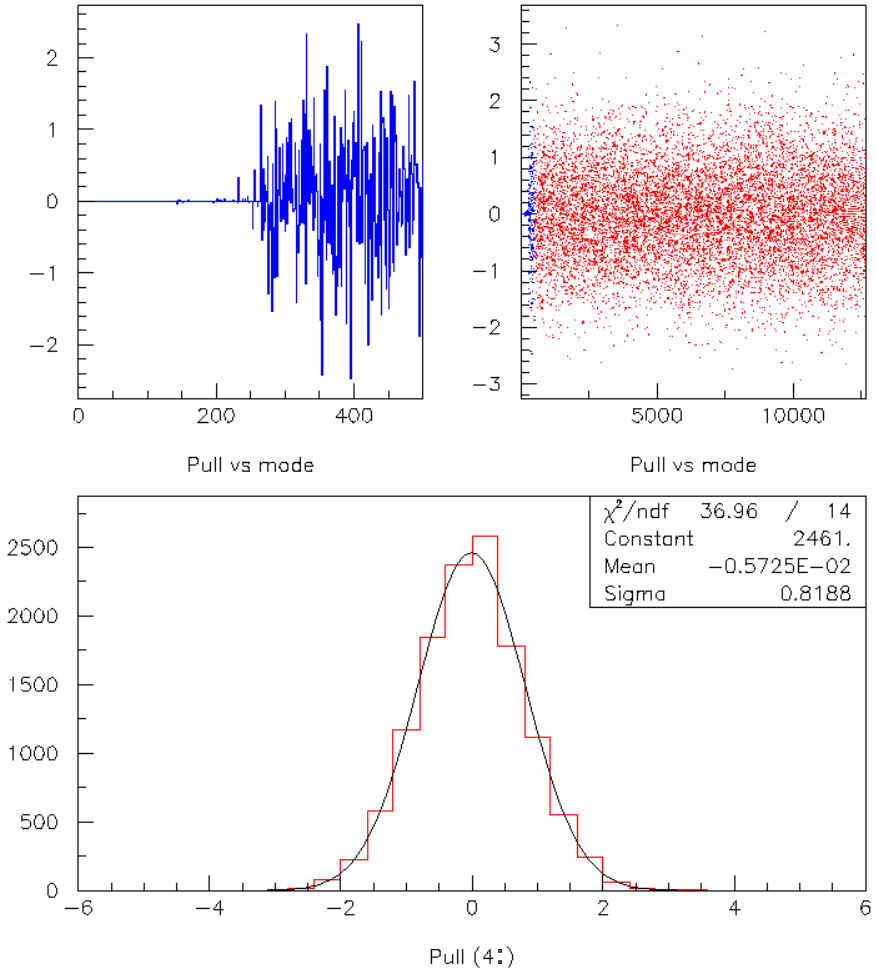




Solve Barrel SCT Problem (12k-DoF):

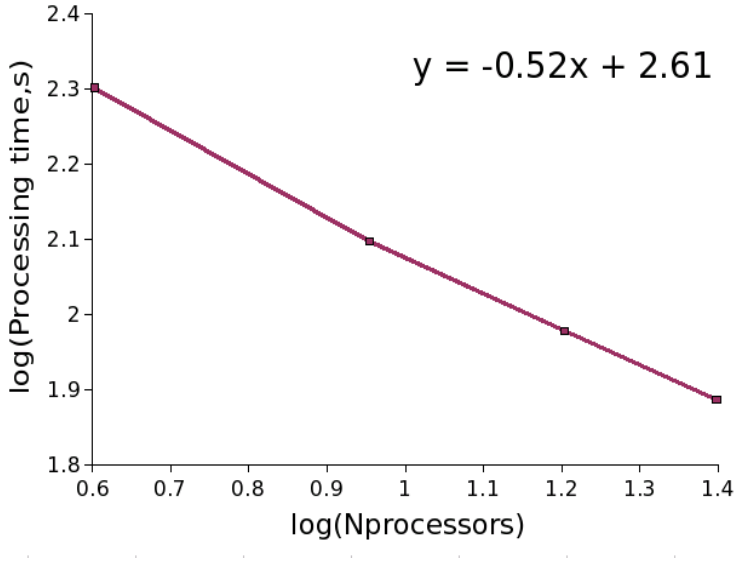
- 1 SCT module remains unpopulated: 6 real spurious modes in matrix as well as some others due to limited statistics.
- This size of a problem can be "solvable" in both single and parallel platforms.
- A factor of ~50 is gained in time performance.

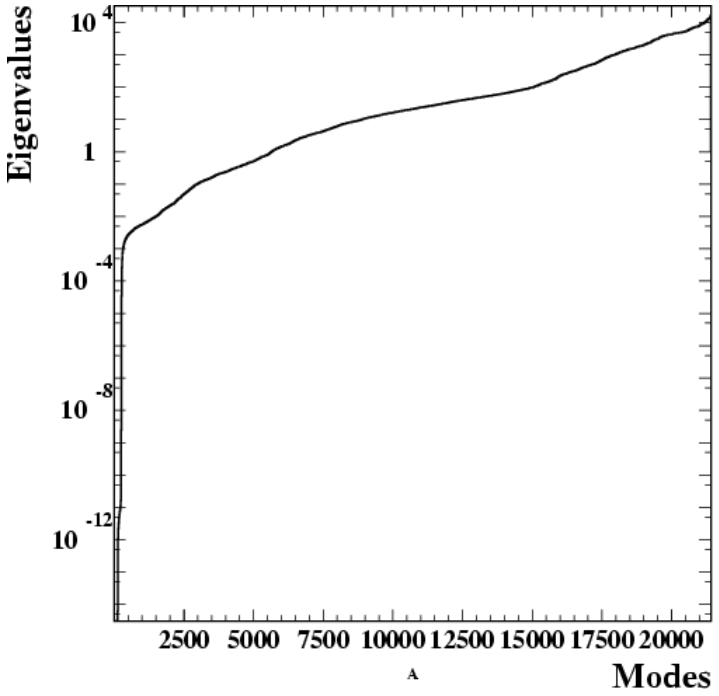




Solve Full Barrel Problem (21k DoF):

- Extended the problem to solve full barrel size.
- A number of pixel modules unpopulated (37), due to coverage limitation in current sample.
- Problem only can be solved on SCARF cluster, unless matrix is written in triangular format (libraries linked with ATLAS software imposes file size limitations of 2GB on the P4 machines)
- Size dependence ~$N^{2.3}$ and N(CPU) dependence ~ $N^{-0.5}$ (using results from our earlier samples with an obsolete Athena release).





Comparison of Platforms for Global χ^2 Solution:

- Parallel-computing allowed for large problem solutions in a very efficient way.
- The time performance looks very advantageous!
- Number of modules in parantheses show the modules not taken into account for alignment.

Nmodule	NDoF	File size(GB) NxN(triang)	Processing time*		
			SCARF NCPU=16 / 4 (2.2GHz, MB=256MB)	Intel PentiumIV	SCARF single-CPU
1030 (0)	6180	0.3 (0.15)	1m53s / 3m	35m (2.8GHz)	39m
2112 (1)	12672	1.2 (0.6)	10m30s / 26m58s	7h34m (2GHz)	5h28m
3568 (37)	21408	3.4 (1.7)	43m43s/ 121m51s	---	**>20 hrs**

* Measurements include the binary file I/O.

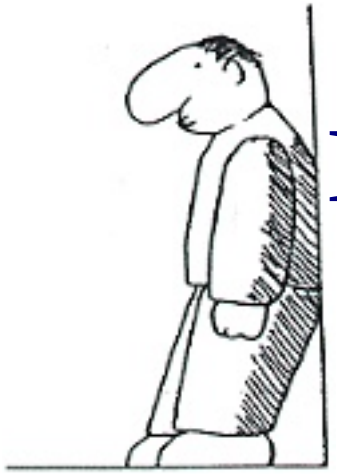
- Also attempted invert these singular matrices. Matrices either found completely singular (as expected) (RCOND=0.0) or RCOND< ε (DoF=6180), and no solution is provided.

Summary & Conclusions:

- We performed tests on using distributed-memory parallel-computing for ATLAS alignment.
- For large scale problems, parallel-computing is very adventageous.
- The performances we obtained are improvements over earlier test results on other platforms.
- Now that we can run on large system sizes and able to obtain final alignment constants, the algorithm and numerical issues will be debugged and understood better.
- ATLAS has plans to port the offline software to 64-bit mid-summer in conjunction with gcc 3.4.4 migration. When this happens, a better infrastructure will be in place for global χ^2 alignment processing.
- We are exploring all possibilities other than we used here (libraries, etc..): this is still work in progress…

Acknowledgments:

- We are grateful to the CCLRC group for their permission to use their Beowulf cluster.
- We are grateful to the OxPP SysOps and Laura Gilbert who is responsible for ATLAS software installations on the OxPP cluster.

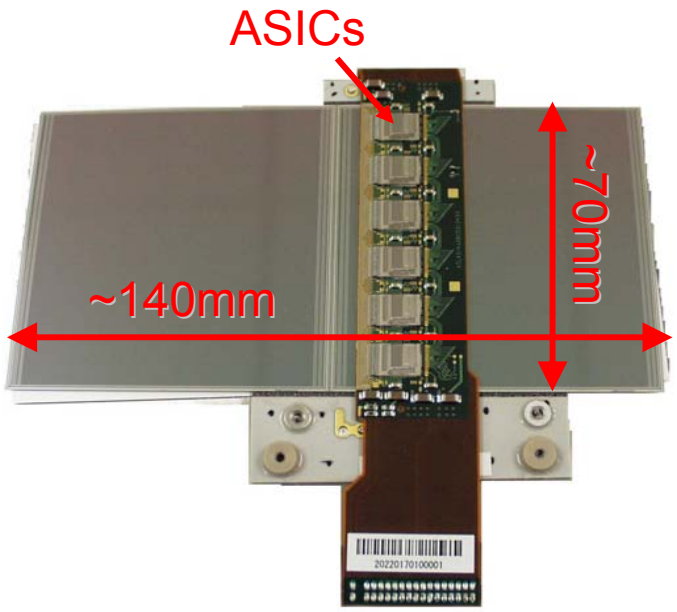


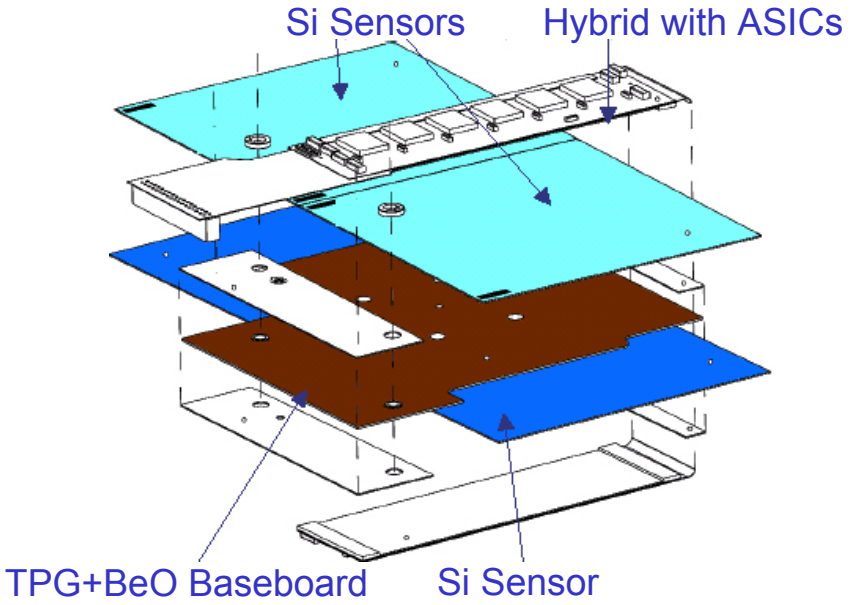
Backup Slides

SCT modules

“Weak” modes

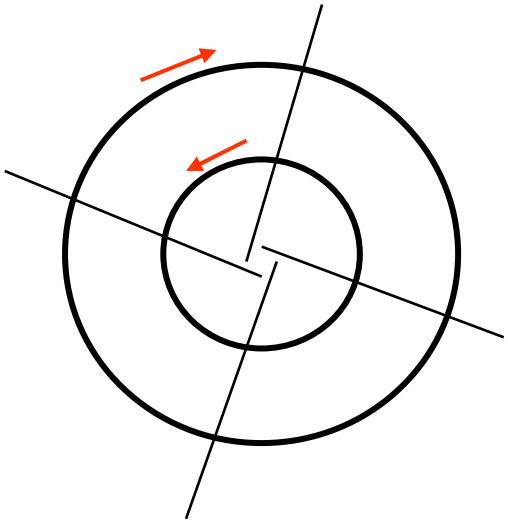
The Building Blocks: Barrel Modules



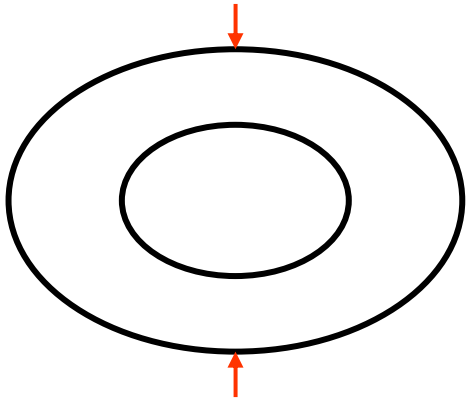


- PIXel detectors provide real 2-D readout.
- Pixels are of size 50×400 μm resulting in 14×115 μm resolution.
- SCT modules are double-sided strip detectors with 1-D readout per side.
- Sensitive strips have pitch of 80 μm giving 23 μm resolution.
- Stereo-angle of 40 mrad gives 580 μm resolution in the other direction.
- Entire tracker is equipped with binary readout.

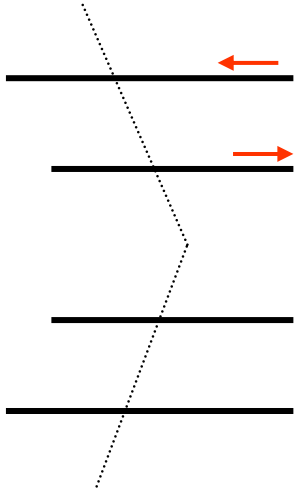
Examples of 'Weak Modes':



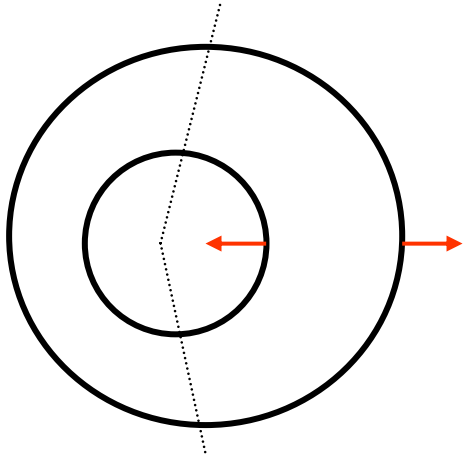
"clocking"
$\delta\phi=\lambda+\beta/R$
(VTX constraint)



radial distortions
(various)



"telescope"
$\delta z \sim R$



ϕ dependent sagitta
$\delta X=a+bR+cR^2$

We need extra handles in order to tackle these.
The natural candidates are:

- Requirement of a common vertex for a group of tracks (VTX constraint),
- Constraints on track parameters or vertex position,
- External constraints on alignment parameters.

η dependent sagitta
"Global twist"
$\delta\phi=\kappa R\cot(\theta)$