Exploiting Parallel R in the Cloud with SPRINT

Published in final edited form as:
Methods Inf Med. 2013 January 14; 52(1): 80–90. doi:10.3414/ME11-02-0039.

Exploiting Parallel R in the Cloud with SPRINT

M. Piotrowski,
EPCC, James Clerk Maxwell Building, The University of Edinburgh, EH9 3JZ

G.A. McGilvary,
Edinburgh Data-Intensive Research Group, School of Informatics, The University of Edinburgh, EH8 9AB

T. M. Sloan,
EPCC, James Clerk Maxwell Building, The University of Edinburgh, EH9 3JZ

M. Mewissen,
Division of Pathway Medicine, The Chancellor's Building, The University of Edinburgh, EH16 4SB

A.D. Lloyd,
The University of Edinburgh Business School, 29 Buccleuch Place, Edinburgh EH8 9JS

T. Forster,
Division of Pathway Medicine, The Chancellor's Building, The University of Edinburgh, EH16 4SB

L. Mitchell,
EPCC, James Clerk Maxwell Building, The University of Edinburgh, EH9 3JZ

P. Ghazal, and
Division of Pathway Medicine, The Chancellor's Building, The University of Edinburgh, EH16 4SB

J. Hill
Applied Modelling and Computation Group, Department of Earth Science and Engineering, Imperial College, London, SW7 2AZ, UK

Abstract

Background—Advances in DNA Microarray devices and next-generation massively parallel DNA sequencing platforms have led to an exponential growth in data availability but the arising opportunities require adequate computing resources. High Performance Computing (HPC) in the Cloud offers an affordable way of meeting this need.

Objectives—Bioconductor, a popular tool for high-throughput genomic data analysis, is distributed as add-on modules for the R statistical programming language but R has no native capabilities for exploiting multi-processor architectures. SPRINT is an R package that enables easy access to HPC for genomics researchers. This paper investigates: setting up and running SPRINT-enabled genomic analyses on Amazon's Elastic Compute Cloud (EC2), the advantages of submitting applications to EC2 from different parts of the world and, if resource underutilization can improve application performance.

Methods—The SPRINT parallel implementations of correlation, permutation testing, partitioning around medoids and the multi-purpose papply have been benchmarked on data sets of

T. M. Sloan **(***Corresponding Author***)** Telephone: +44 (0) 131 650 5155 tms@epcc.ed.ac.uk Fax: +44 (0) 131 650 6555 .

Europe PMC Funders Author Manuscripts

various size on Amazon EC2. Jobs have been submitted from both the UK and Thailand to investigate monetary differences.

Results—It is possible to obtain good, scalable performance but the level of improvement is dependent upon the nature of algorithm. Resource underutilization can further improve the time to result. End-user's location impacts on costs due to factors such as local taxation. Conclusions: Although not designed to satisfy HPC requirements, Amazon EC2 and cloud computing in general provides an interesting alternative and provides new possibilities for smaller organisations with limited funds.

Keywords

Genomics; Computing Methodologies

1. Introduction

There are few other fields in science that have developed so rapidly in the last few decades as Genetics and Biotechnology. With the advance and commercialisation of DNA Microarray devices and systems [1] the process of reading multiple gene expression levels or single nucleotide polymorphisms (SNP) can be done in a highly efficient parallel manner, leading to exponential growth of available microarray data. This data and technology allows the scientific community to better understand the nature of genetically caused anomalies and identify gene signatures that might correspond to diseases like cancer [2, 39]. Experimental microarray data, where published, is in most cases stored in publicly available repositories such as The Gene Expression Omnibus (GEO) [3] or The ArrayExpress Archive [4] making them accessible to many more scientists and projects. The number of microarray samples that researchers have access to numbers hundreds of thousands. This allows significant refinement of the outcomes of many Genomics research projects, as long as scientists have enough computing power to process these huge amounts of data. This greatly increased the rate at which data on biological systems is generated and has led to corresponding challenges in analyzing the data through advanced computational techniques.

In the last few years there has been a further technological breakthrough in the field of DNA sequencing. The current next-generation massively parallel DNA sequencing platforms like the 454 Pyrosequencing system or Illumina Genome Analyzer [5] have accelerated the sequencing process and reduced the cost of DNA sequencing by several orders of magnitude, breaking the monopoly of major genome centres for large scale genomic research. With this affordable next-generation technology even small research groups are able to conduct experiments such as studying the nature of the physical and behavioural inheritance process by sequencing the DNA of more complex organisms involved in a matter of days rather than months. It is expected that with the current pace of DNA sequencing technology development, personal genomics will soon become a common patient diagnostic practice.

These new technologies are opening up fascinating opportunities in the field of biology but they also require adequate computing resources and methodologies. For example, a single sequencing run can produce about 1TB [6] of data and usually a full experiment requires 20 or more sequencing runs. Processing and analysis of data of such a size means billions or trillions of operations need to be executed. It has therefore become obvious that it is not possible to fully exploit this new technology without overcoming the software and infrastructure limitations that are often present for smaller research facilities.

One way to overcome these barriers is to take advantage of High Performance Computing (HPC) solutions, such as SMP (Symmetric Multi-processing) clusters, GPUs (Graphical

Processing Units) or even multi-core workstations. However, investing in HPC infrastructure may require substantial funds, staff effort and expertise. Hosting and maintaining a purpose-built scientific cluster, which in most cases is often underutilised, is not the most flexible solution for ventures with limited budgets. In such cases Cloud computing seems to be a more elastic and affordable alternative. Whether it really is more efficient and cost-effective in the long term still needs to be proven.

In addition, even the most powerful computing cluster is useless without proper software. Bioconductor [7] is one of the most popular tools for the analysis of high-throughput genomic data. It is free, open source and distributed as a set of add-on modules for the R statistical programming language [8]. R itself is a high-level, interpreted language that has no native capabilities for exploiting multi-processor architectures. The Simple Parallel R INTerface (SPRINT) [9] is an R package that offers both a parallel functions library, optimised for processing huge data sets, and an interface for adding parallel functions to R. The aim of the SPRINT package is to allow R users to use parallel versions of the most common computationally intensive, genomic analysis functions, without the need for parallel programming skills and experience. A number of parallel functions (namely correlation, clustering, rank product, permutation test, bootstrap, random forest classifier and the apply method) are available in SPRINT. The SPRINT package has been tested on several HPC architectures including the UK's national supercomputer service HECToR and has exhibited very good performance and speedup up to 512 processors ([10], [11], [12]) The SPRINT package has therefore proven to be an efficient parallel solution for R users on dedicated HPC infrastructure.

2. Objectives

In the context of rising interests and applications of Cloud technologies, the objective of the work reported here is to investigate the performance of SPRINT when it is executed within a Cloud environment in order to determine if Cloud computing can offer a viable alternative for those organisations who have reached a limit in their existing computational infrastructure for genomics data analysis but do not wish, or indeed, have the necessary funds, time and expertise to purchase, build, maintain and exploit a dedicated HPC resource. Although, SPRINT is designed to work on HPC clusters, it should be possible to run it on any parallel architecture that supports MPI [21] and has access to a shared disk space. To test this, Amazon's Elastic Compute Cloud (EC2) was chosen as the IaaS (Infrastructure as a Service) provider for the Cloud environment simply because at the time of writing it is the most popular and widely accessible service worldwide. This paper therefore documents the performance of SPRINT on EC2, the steps needed to set up a dedicated cluster within the virtualized environment and how to deploy the required software and data upon it for running SPRINT-enabled analyses.

Despite the generic advantages Cloud computing offers, these infrastructures still have limitations regardless of the software used. One of these limitations is the variability of the platform which can appear in many forms. Due to the global nature of Cloud computing, regional dependencies can exist where it may be more advantageous to submit applications to the cloud environment from one part of the world than another. This variability can have a substantial effect where, for example, the operational costs of one laboratory may be greater than that of another based in a different country using the same technologies. SPRINT has therefore been used to illustrate the magnitude of cost disparities between submitting computational jobs from both the UK and Thailand.

Regardless of country, the computational job's configuration can affect the operating costs as these depend on the amount of resources reserved in a Cloud. By exploring the effect of

underutilization, this work will also illustrate how to improve an applications performance on EC2 via underutilization of resources, i.e. committing more resources than the application requires in order to improve the reliability of the performance and hence the quality of service to the end user. This in turn can lower costs due to the lowered completion time of the application, however the rate of underutilization is application dependent. Hence common, vital computational jobs will give differing performance and cost results simply due to their chosen cloud configuration. This paper reports on experiments undertaken to discover the effects of end-user location and the configuration they may use to run their computational job. Since the results produced are likely to affect a large population of potential cloud applications, this can give existing and new research organisations the required information on how to best exploit the use of Amazon EC2 as well as other cloud provider services according to their research priorities.

In summary, in this paper we report on:

- the installation and configuration of R and SPRINT in a cloud environment, specifically Amazon Elastic Cloud (EC2) [13];
- benchmark results collected for a number of SPRINT-enabled R functions on these Amazon EC2 resources;
- using SPRINT to probe whether regional differences between the UK and Thailand exist in monetary terms on the cloud; and finally,
- using SPRINT to investigate the merits of resource underutilization on the Amazon EC2 cloud.

3. Background

3.1 In the Clouds

The costs of building and maintaining a scientific computing cluster for processing large data sets gathered by modern genomic analyses are high. Not only is there the cost of computing hardware, there is also the cost of software licences, operational costs such as energy, additional staff and potentially software consultancy to port existing data processing workflows. Organisations dealing with genomic research however, increasingly require access to high-end computing resources. Those with limited funds are more likely to consider moving to a computing service model offered by many Cloud providers. This is therefore a potential new requirement of software or platform solutions.

There are a number of projects that attempt to utilise Cloud computing for genomic research. One example is the BLAST [14] library, one of the most widely used and computationally demanding tools for rapid sequence comparison. Several parallel implementations of the library are being developed on different Cloud platforms including Amazon EC2 and Microsoft Azure [15], [16]. The Genome Analysis Toolkit (GATK) [17] is another very interesting analysis tool for next-generation sequencer data. This is a programming framework written in Java that is based on the functional programming paradigm of MapReduce [22]. The advantage of MapReduce is that it can scale very well on large number of nodes. However, a major limitation is that the map and reduce steps must not depend on each other thus narrowing down the number of problems that could be solved using this approach.

3.2 Amazon Elastic Compute Cloud (EC2)

Amazon Web Services (AWS) offer their Elastic Compute Cloud as an Infrastructure as a Service where virtual machines (or instances) are provided with access to raw resources such as storage and networking. This enables users to rent scalable virtual computing

resources, available in multiple locations across the globe, divided into different Regions and Availability Zones. Once such a resource is rented, a user can choose and instantiate a preconfigured system image, called an Amazon Machine Image (AMI), with a selected operating system and a set of desired software that is already installed. Once booted, such a virtual machine, called an instance in the Amazon nomenclature, can be accessed via any ssh client after providing the appropriate access key. A number of such instances can be grouped into a computing cluster.

Instances are deployed upon data centre machines within a specific Region chosen by the cloud user. Regions are defined as separate geographical areas within which, multiple Availability Zones exist that are distinct areas that aid in replication and increase fault tolerance. For example, if an instance fails in one Availability Zone, an application can still operate provided that another instance is running in another Availability Zone. However costs for computation, storage and network communication can differ in each of these Regions.

As a result of the disjointed infrastructure, several data-transfer mechanisms exist: Internet Data Transfer (IDT), Regional Data Transfer (RDT) and Availability Zone Data Transfer (AZDT). IDT describes data transferred from within an instance (or any Amazon Web Service) to a location outside the Region, and vice versa. RDT describes data transferred between instances in different Availability Zones within a Region; its cost equivalent in all Regions. Finally, AZDT is data transferred between instances within the same Availability Zone and is free.

3.3 SPRINT – Simple Parallel R INTerface

SPRINT is one attempt to incorporate parallel capabilities into R. Schmidberger et al [18] provide an overview of other packages that offer tools for enabling parallel execution of R scripts. However, most of these packages only allow the parallelisation of trivial problems, where no data dependency occurs. This includes the more recent parallel R package available in R version 2.14. This usually means that each process taking part in the computation receives a portion of data and operates independently of other participating processes in a so called “task farm” approach. The Rmpi package [19], which is a R wrapper around Message Passing Interface (MPI) [21], allows a fuller exploitation of parallel programming techniques. However, this requires knowledge and experience in parallel programming. What makes SPRINT different is that it offers easy access to HPC, yet hides the complexity of parallel programming whilst still offering access to functionality that operates on complex data dependent workflows. It contains a library of selected R functions that have been parallelised and allows the addition of parallelised functions to R by the community. To ease usability, the interface of the parallelised functions is deliberately kept as similar as possible to the original functions. For example, a call to a serial correlation function cor(MtxA, MtxB) becomes simply pcor(MtxA, MtxB).

The core of the SPRINT framework is implemented in the C programming language and the MPI parallel programming interface whilst the user interface is written in R. The package provides an HPC harness that allows R scripts to run on a number of parallel architectures from HPC clusters to multi-core desktop machines. SPRINT should be able to run on any Cloud cluster as long as all computing nodes have access to a shared disk space. A more detailed view of the architecture of the SPRINT runtime is presented in Figure 1. The SPRINT architecture acts as an HPC harness managing a number of processors that can be assigned different tasks or put to sleep while the sequential part of the R script is executed.

In Figure 1 Figure 1a job starts by requesting a predefined number of processors and initialising the R runtime on all the acquired nodes. The SPRINT library is loaded and

initialises the MPI environment. Based on a process identifier, SPRINT distinguishes between the master and worker nodes, which follow different execution paths. All worker nodes immediately after being initialised, enter a loop waiting for command code sent from the master process. After the SPRINT library is loaded the R script execution is continued only on the master process. When an R SPRINT-enabled function is called in the R script, execution control is passed from R to SPRINT whereupon the master process wakes up all the worker nodes and broadcasts a function signature that will be invoked on all processes. Depending on the function, the data is distributed between processes and computation continues in parallel. The fact that MPI is initialised in the R memory space means the R runtime environment can be accessed on all the processes. This allows the handling of native R objects in C and most importantly means that R expressions can be evaluated from C. This feature provides flexibility when adding new functions to SPRINT meaning that parallel SPRINT-enabled functions can either consist of a complete parallel re-implementation of the function or utilise the existing serial R implementation of a function within a parallel harness. After all the computation is completed on the worker processes, results are sent back to the master process and returned to R. This is repeated for all parallel SPRINT functions present in the user's script. Before exiting the R runtime the **pterminate** function is called on the master. This shuts down the SPRINT framework, by sending an appropriate command to all nodes and so terminates all the worker processes.

4. Methods

4.1 Setting up an Amazon EC2 cluster for SPRINT

The process of creating an MPI cluster on Amazon EC2 is not simple and requires some experience with installing and configuring software under UNIX. There are public images available that provide some support for creating an MPI cluster. For example, the Bioconductor [7] project provides a public Amazon Machine Image (AMI) – a pre-configured operating system allowing others to instantiate virtual machines from the image – with a pre-installed version of R, Bioconductor packages, as well as the MPICH2 library [23] and scripts that after some modification will allow users to launch additional images and run MPI jobs. However, at the time of writing, this Bioconductor AMI can only be used in Amazon's US-East-1 Region. Moreover, configuring an AMI manually enabled control of every aspect of the installation so allowing the authors to investigate the optimum configurations for SPRINT.

We therefore set up our own AMI with the Ubuntu 10.04 Operating System deployed on the Standard On-Demand Extra Large instance type (m1.xlarge) with 15 GB of physical memory, and 4 virtual cores with 2 EC2 Compute Units each. According to Amazon [20], one EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. The instance interconnect is unidentified and described only by a high I/O performance parameter that indicates the instance has a larger allocation of shared I/O resources. Setting up our own instance from scratch also allowed full control of its configuration so that it could be optimised for SPRINT usage and to assess the overhead in doing so.

At the time that the benchmarks were run using the above instance, the Amazon HPC instances were not available. Whilst these HPC instances did become available before the article was submitted this was only in the US East Region. Benchmarking SPRINT on these HPC instances is therefore future work particularly as they become more widely available.

4.2 Benchmarking

Three representative SPRINT functions have been benchmarked to investigate how these perform in the Amazon Cloud environment.

The data used in the benchmarks was gleaned from Agilent Mouse Oligo Microarray V2 (G4121B). Each array allows measurement of 22,393 probe transcripts' mRNA level. The dataset contains 75 biological samples hybridised to 75 arrays, divided into 3 biological conditions, with each condition comprising 25 of the arrays, and each array within a condition measuring one of the 30min interval time points from 0min to 720min. The data was background corrected and normalised in R. Due to security concerns surrounding Cloud computing, the data was made anonymous with all metadata information stripped out and the order of rows randomised [37]. Cloud security and therefore its usability for analysis of vulnerable genomic data is a serious problem and a topic for another paper.

To conduct the benchmarks, three functions have been selected whose, implementation has required different aspects of parallel programming. Each function's behaviour is different when it comes to I/O operation intensity, data distribution and inter-node communication. This should allow one to better understand and identify the strengths and weaknesses of a computational Cloud for HPC genomic analyses. The selected functions are the SPRINT parallel implementations of the R Pearson correlation function ie. **pcor**, the partitioning around medoids function ie. **ppam** and the multi-purpose **papply**. All three functions were included in a single R script with the results passed from one function to another to form a complete workflow. The numbers of genes selected for the experiment from the previously described data were 14,821 and 29,642. The larger dataset was created by combining filtered data for two different conditions. Two datasets were used in the benchmarks to examine how SPRINT performance on the EC2 was affected by an increase in size of dataset. The time required for the initial microarray data load is not included in the results reported below, only the total time spent in each function. The timings were repeated 5 times and the mean value used to calculate the speedup. Due to the shared nature of Cloud environments, the results obtained are more variable than on an infrastructure where one has exclusive access. This was particularly noticeable in the longer running benchmarks where there was far greater variability compared to the quicker running results. Due to memory constraints when analysing data sets of this size, it was not possible to run the full workflow using the sequential R implementations of these functions. The presented speedup is therefore relative to the execution time on two processors.

The R script starts by calculating a distance matrix of expression profiles between genes using the **pcor** function. The genes that show similar expression levels are clustered into 24 groups by **ppam**, the parallel partitioning around medoids function. The size of a distance matrix is 1.6 GB or 6.6 GB depending on the number of genes used in a run. Finally, just for benchmark purposes, a dummy function sub-sampling the expression levels in each group is applied using **papply**.

4.3 The Effects of End-User Location

Regional cost differences, dependent on where an instance is deployed, are known to exist in Amazon EC2. We, however, chose to test whether differences exist dependent on a user's job submission location and whether they are significant enough to make it advantageous for a business or research institution to submit a job to Amazon EC2 from one location than any other. To examine such differences, we submitted a job from two distant and distinct locations in the world, the UK and Thailand.

We ran our distributed application over two Standard On-Demand Large EC2 instances (m1.large) each with 7.5 GB of main memory, and 2 cores each with 2 EC2 Compute Units. These instances were located in the US East Region within the us-east-1b Availability Zone. The job ran the **pcor** function processing a randomly generated dataset consisting of 11,000 genes and 321 samples building upon previous work of benchmarking **pcor** on both the Edinburgh Compute Data Facility (ECDF) Cluster [9] and HECToR [35]. The principal focus was to create an experiment that was fair and consistent across runs, hence a collection of scripts was created to automatically instantiate instances, setup the experiment, run the experiment and teardown instances allowing the experiment to be run consistently by multiple researchers regardless of whether they were in the UK or Thailand. In collaboration with Dr Sornthep Vannarat, Head of the Large Scale Simulation Laboratory from the National Electronics and Computer Technology Center (NECTEC), the experiment could be performed in Thailand. Once the computation was complete, the cost and resource usages were obtained from the Amazon EC2 invoice and resource Usage Report (a simple xml or csv file) respectively.

To ensure the results were valid, confirmation was required that Dr Vannarat's Amazon EC2 account was tied to an address in Thailand, otherwise if not, different charges could be seen. Another option, later ruled out, was to submit a job via a Thai Virtual Private Network (VPN) from the UK, however charges would still be tied to the Amazon EC2 UK account, incurring UK costs as explained in the results section later in this paper.

4.4 Resource Underutilization

The idea of reserving more resources, while only using a small percentage of each to potentially increase performance, is counter-intuitive. This may not occur for other platforms, however, because of the time-shared nature of cloud computing, resource contentions will arise. To provide evidence of this, we performed an experiment running SPRINT in two different cloud configurations running a number of EC2's Standard On-Demand Large instances. Exploring underutilization using Large instances allows us to uncover the effects of not only the consequences of CPU contention but also how communication between them affects the computation. Using Amazon's dedicated HPC instances, such as Cluster Compute instances, would have only allowed us to investigate CPU contention, as fewer of these instances would be required. Testing SPRINT on Amazon's HPC instances is future work. The functions used were the **pcor** and **pmaxT** parallel functions where the latter is computationally intensive as opposed to the memory bound function **pcor**, and processes a randomly generated dataset consisting of 6,102 genes and 76 samples per gene with a permutation count of 150,000. The **pcor** function used the same input data set as the previous experiment on End User Location.

The two cases used to explore underutilization are defined below.

- Case 1: processes are run on separate instances using only one core on each, ie, half of the processing capacity available. For example, 4 processes would run over 4 instances consuming only one core of each.
- Case 2: processes are run on each core of an instance, fully utilizing the instance's computational resources. For example, to instantiate 2 processes, we would deploy 1 Large EC2 instance consuming both its cores as opposed to running 2 instances in Case 1; the use of 4 cores would be spread over 2 large instances etc.

5. Results and discussion

5. 1 Cloud Set-up on Amazon EC2

Before deploying all the necessary software on Amazon EC2, the storage mechanism needs to be chosen. Amazon's Simple Storage Service (Amazon S3) is designed to be a reliable and scalable Internet storage that provides a web service interface. It is possible to access Amazon S3 storage from the multiple EC2 instances that are required to run SPRINT in the EC2 Cloud. However, whilst the S3 performance may be enough to handle even the busiest website traffic, it is not designed to work with I/O intensive HPC applications. The Amazon Elastic Block Store (EBS) is a new type of storage that is designed specifically to work with Amazon EC2 instances. EBS behaves just like an external hard drive that is attached to our virtual machine. It is designed to have significantly lower latency and higher throughput than Amazon S3. There have been several attempts to benchmark the S3 versus EBS performance [24]. These show very variable results that depend hugely on the network traffic and Cloud load. Generally the conclusion from these is that EBS is much faster for frequent reads and writes. SPRINT is typically used to process very large data sets, the analysis of which would be intractable on standard IT infrastructure. Disk performance is therefore vital for some of the SPRINT-enabled R functions programs to scale on large computing clusters due to the amount of I/O needed to manipulate these large data sets.

Amazon EBS allows the creation of volumes that can be mounted as devices by EC2 instances, this works well as long as there is no need to share data concurrently. The problem in such circumstances is that it is not possible to attach a single EBS volume to multiple instances at the same time. This situation is required to run SPRINT. However, this limitation can be overcome with widely available, common solutions. The Network File System (NFS) enables direct access to files located on remote systems that are connected to a network. Setting up an NFS server on the master node allows us to expose the EBS drive to other instances in the cluster. The configuration process is similar to that of a traditional cluster with one exception, the security settings. It is important to configure your security group accordingly, so only the machines you trust will have access to the files that you are sharing. This is an important consideration when dealing with sensitive or confidential data.

The next step is to install all the required software. SPRINT is recommended for running with the MPICH2 implementation of the MPI-2 library. R and the SPRINT package are installed on to the master node. Finally the last step is to configure the MPICH daemon by modifying the appropriate MPI configuration files. The file **mpd.conf** contains a secretword parameter that needs to be equivalent on all nodes. The **mpd.hosts** file contains the list of all nodes in the cluster and the number of available cores and so needs to be modified. Once this is finished a snapshot of the AMI instance can be taken and used as a template to create additional nodes. It is important to modify the MPI configuration files on the additional nodes. The **mpdboot** command can now be used to start the cluster and the **mpdallexit** command will shut it down. The **mpdtrace** command lists all nodes in the cluster and is a useful command to test if the cluster has launched properly. At this point it should now be possible to run a parallel job on the EC2 cluster using the **mpiexec** command.

5.2 Benchmarking

The benchmark results shown in Figure 2 are for the dataset with 14,821 genes and 75 biological samples. Here all the functions show relatively good speedup as the number of the processes increases. Petrou [35] and Piotrowski et al [11] have previously shown that, with a suitably sized dataset, **pcor** and **ppam** exhibit near linear scaling on up to 32 processes on a Cray XT supercomputer consisting of 1416 compute blades each with four quad core processor sockets where the CPUs were AMD 2.3 GHz Opteron chips with 8 GB of

memory. These blades were connected via the proprietary CRAY SEASTAR2 interconnect with access to high performance parallel I/O disk storage as opposed to network attached storage. In Figure 2 the speedup of **pcor** and **ppam** between 2 and 4 processes is definitely weaker than when moving from 4 to 8 processes i.e. moving from 1 Extra Large instance to 2 instances. This could be explained by the fact that when the program is run using 4 cores all the processes have to compete for the same resources on the same instance, e.g. access to same system for reading/writing to disk, thus less performance is gained. However, as mentioned previously, exact interpretation of results can be difficult due to the variability in performance as a result of the shared nature of the EC2 infrastructure. The speedup for both **pcor** and **ppam** tails off for 32 processes. The **pcor** function replicates the input data on all processors before the computation is started. For small data sets and large numbers of processes the time required for sending the input data can overshadow the computation time. As we will see later Figure 2 is the only scenario where we managed to get some speedup from the **ppam** function with a small number of processes.

The **papply** function shows the most stable speedup up and scales up to 32 processes (ie. 8 Extra Large instances). Scaling of the **papply** method depends on the actual function that is passed as an argument. The in-lined function used in the benchmark resamples the data N times where N is the length of the data set and performs some simple computation. This is an ideal scenario for parallelisation since inter-process communication is minimal and the cost of distributing the data and gathering the result is small compared to the computation time.

The benchmark results for the data set with 29,642 genes are shown in Figure 3. With the increased data set size, the **pcor** function exhibits much better parallel efficiency and has almost linear speed-up i.e. the performance nearly doubles for twice the number of processors. The function scales well with the size of data and together with the equally well performing **papply** method, shows that shared utilisation and variable performance of the Amazon EC2 interconnection does not affect the overall performance of such algorithms, where the ratio of computation to communication is high, provided there is sufficient data to process.

The partitioning around medoids function behaves differently. It performs a lot of I/O operations and its parallel implementation requires that the participating processes exchange data many times with each other. Whilst the **ppam** function enables R users to process data larger than the available memory it does so by utilising file mapping and this results in frequent disk access. This requires a fast interconnect and efficient I/O in order for it to scale to higher numbers of processes. Clearly Figure 3 shows that the solution adopted with the relatively fast EBS exposed to all nodes by the NFS on the master node is probably not efficient enough and is the main performance bottleneck. This can be seen when comparing these results with previous benchmarks conducted on a local cluster that achieved much better scaling [11]. These were performed on a shared memory cluster with 8 dual-core processors, where each process had simultaneous access to data stored on a disk. Another attempt to parallelise the PAM algorithm using Amazon EC2 and the MapReduce approach [22] showed that not all parallel algorithms fit the simple divide and conquer approach that is the most efficient strategy when utilising and scaling up on Cloud infrastructures. However it is important to bear in mind that the results do indicate that the cloud infrastructure can be used to process a size of dataset untenable on a local resource due to memory constraints.

Similarly, whilst the SPRINT **ppam** does not scale well on EC2 its performance is still significantly better than the original serial R implementation even on a single node. Figure 4 compares running on EC2, both the original serial R implementation and the optimised and

parallelised **ppam** available in SPRINT. Although no speedup is achieved increasing the number of processors, the time required to cluster the data on 2 processes using **ppam** is approximately 8 seconds while the time spent on a single processor using the **pam** function required 752 seconds to complete.

5.2 End-User Location

From Table I we can see that the total cost of running two large instances for the same period of time, including other associated costs such as data transfer and I/O requests, is more expensive when the job is submitted from the UK than in Thailand. This is caused by the level of taxation within the two countries where the UK charges Value Added Tax (VAT) at 20%, hence an increase in $0.42, whereas Thailand charges no taxes. For small and cash-flow sensitive businesses and research institutions, this difference may have a significant impact on growth, for example, a business spending $4000 on cloud costs per month. For one year of use, the contribution to VAT at 20% would be $9600; more than two months of cloud usage. The impact of VAT differentiation on market share has been studied across Europe and found to have a highly significant impact, for example a 10-15% reduction in price leading to predicted increases in demand of 70% [36]; VAT is therefore one of the many important choices a global organization must factor into account when choosing where to operate from and provides a significant incentive for businesses adopting a cloud model to migrate data to regions where running costs are lower.

Note that total running costs must take into account regional cost differences and whether tax is charged; prices specified by Amazon are not taxed however if VAT is charged in your country, additional costs apply. In such cases, taxes are calculated based on the address of a user's account and so the service might be outsourced to a tax-free region, reducing the direct cost of the final service offered to the consumer, making it possible to price any service provided with more sensitivity to the competition.

Alongside monitoring experiment costs, the amount of resources used was also obtained. Table I shows significant differences for I/O requests and Internet Data Transfer (IDT) inwards. 70MB's of extra data transferred to the US Region from the UK was recorded, accounting for an addition of $0.01 compared to the Thailand run. This is likely caused by data retransmissions when transmitting data to the instances and installing the necessary packages for the SPRINT experiment to run. This table also shows 37,580 fewer recorded I/O requests from the UK, accounting for $0.01 less than its counterpart, hence levelling the costs of both runs. Why such a phenomenon occurs is likely to result from EC2's underlying storage reading and writing mechanisms, however experimentation to uncover the exact cause of this variability is future work.

The structure of Amazon EC2 may also bring limitations to those with applications that require the highest performance available or those with data privacy issues. In the former case, an application that transfers large amounts of data from an external web service in the UK to run computations, would perform better if the instances were located in Amazon's European Region as opposed to the US or Asia Region, due to the locality of the data. In this case, we may see an increase in data transmission and computation speeds. Hence an organisation must perform a full analysis of their computation job to determine it characteristics - for example, whether data must be transferred long distances - allowing a decision to be made on whether lower costs or higher performance is of utmost importance.

In the latter case of data privacy, organisations may be disadvantaged by regulations from countries within their continent that do not have an Amazon Region present (e.g Africa or Australia) if no cloud or virtualisation based providers operate locally. A business or research institution with sensitive data may not be allowed to use computational services

residing in other Regions, hence reducing its competitiveness with others where a cloud service present. These aspects, dependent on a business's location factors, must be taken into account in its start-up stages and must be monitored throughout its lifetime to increase its competitiveness within its market.

5.3 Resource Underutilization

Figure 5 shows the results from the two cases used to explore resource underutilization. This Figure shows the obtained completion times according to the number of processes/cores used on each platform. We see that Case 1 and Case 2 provide different times where the former is slower running on 2 cores over 2 instances as opposed to 2 cores on a single instance. This is attributed to the network communication involved between the instances but also the network's performance. However for 4 cores and above, Case 1's completion time is faster for both functions giving two possible explanations: (i) either the larger amount of resources available overall gives a greater speedup than Case 2 or (ii) using both cores of an instance actually degrades performance as opposed to using the same number of cores spread over the same number of instances to cores.

Upon investigation, the former explanation can be disregarded. The **pcor** function and its parameters, did not strain the resources available of even a single instance, hence increasing the resources available overall would not decrease the function's completion time, ruling out why Case 1 performs better when the number of cores are greater than two. This also suggests that Amazon EC2's network performance, which has been found to be poor in comparison to HPC platforms ([25], [26], [27]), offers better performance than that of resource sharing on the same physical server, in this case. Therefore to increase performance, avoiding contention to access shared resources such as CPU, memory and networking, one can instantiate a greater number of resources. In this case, we see that using 50% of the resources gives greater performance than full utilization. This may increase costs, however [28] found that costs can actually decrease if the number of resources reserved and utilization levels are correct. For small cash-flow sensitive businesses, reducing the time and money consumed while running jobs, should be of great importance. For example, the 70 second time difference between the 4 core scenario of running **pmaxT** in the two different configurations, could increase running costs by $0.68 per run via Case 2. If this job were repeated multiple times per day each day, the money potentially saved via a Case 1 configuration, could be substantial after the period of one year. Hence, this raises interesting research questions on whether businesses themselves should spend more to find their optimal job configuration to lower costs overall, over a longer time period.

6. Related Work

A number of different researchers have performed detailed examinations into the performance of the Amazon EC2, some focussed solely on performance variability. However we must not forget other cloud solutions exist, such as Google AppEngine, Microsoft Azure and Rackspace, to name a few. While Amazon EC2 may be today's most popular cloud, analysing the performance and cost variability of other providers is essential.

Iosup et al [29] examine the long-term performance variability of EC2 paying particular attention to the variability of many AWS services that are connected to EC2 such as SimpleDB and the Simple Queue Service. They also consider AppEngine's Python environment showing the completion time differences over the period of a year, as well as the variability of other related services, such as Memcache and Datastore.

Schad et al, [30] investigate the variability of EC2's CPU, I/O and network performance via micro-benchmarks. They show that small and large instances suffer from large performance

variations, which may be partly caused by the underlying hardware an instance is deployed on, such as the AMD versus Intel processors. They also show variations exist between Availability Zones, both in terms of network and CPU performance. A further study would be required to determine the optimum hardware setup and Availability Zone to deploy a computation job in order to boost profits.

Performance variability has also been evaluated in relation to underutilization. Iakymchuk et al [28] show that considerable improvements in speed, of up to two orders of magnitude, can be explained though under-utilization, by investigating CPU and cache contention. They also show that virtualization contributes little to performance degradation, as does [26], and that EC2 nodes can perform as well as nodes found in current HPC platforms. However [26] concludes that, overall, the current state of the cloud cannot be compared to HPC platforms. This position is widely acknowledged [32], however Hill et al [31] demonstrate that EC2's performance can be compared to a commodity Gigabit Ethernet-based cluster, making any conclusion premature. Indeed, the comprehensive study by Armbrust et al [33] discussing a wide range of issues regarding cloud computing, notes that the top 10 obstacles facing cloud platforms include unpredictability with reference to I/O performance.

7. Conclusions

Cloud computing provides an interesting alternative and provides new possibilities for smaller research labs with limited funds for computing infrastructure. In this paper we have demonstrated that a computing cluster on the Amazon EC2 can be set up and used to run HPC software for genomics analysis. The process of installing on and configuring the EC2 cluster and all the required software components is not trivial. However, there are a number of ready to use Amazon Machine Images with preinstalled MPI libraries and scripts that automate the cluster instantiation process. The resulting benchmarks of the SPRINT software on the EC2 cluster has revealed that problems that are easy to parallelise, where computation can be easily divided into smaller tasks with minimum amounts of inter-process communication, should scale very well in a cloud environment.

Due to I/O performance, unspecified physical location of nodes and the interconnect bandwidth being shared with other users; a cloud environment is better suited to those problems that can be tackled using MapReduce or similar approaches than those that require fine-grained parallelisation. There is currently no guarantee that resources provided by the Amazon EC2 are in the same physical location. They are located in the same availability zone but may be scattered among several data centres within that zone. The resources like interconnect and data stores are also shared between other users. More complex applications that require frequent disk access or inter-node communication may require dedicated clusters to perform and scale well in the Cloud. These services are becoming available [34] but at the time of writing are only available in the United States.

By running the same application from two distinct parts of the world, we showed that contrasting costs emerge due to the local taxes employed by the country, adding a new dimension to the flexibility provided by a cloud solution that may impact competitiveness and hence new business start-ups. We also discussed that the competitiveness of some businesses may be limited due to the increased performance they require from the cloud or from data privacy laws restricting where data can be sent for data processing. We then examined some performance aspects of EC2, especially with respect to job configuration where the underutilization of resources can actually increase overall performance, similar to [28] who explore this point further by showing that costs can also decrease. Obtaining the correct configuration for optimum performance and cost is an ongoing problem for researchers and practitioners as it will affect a business's long-term sustainability in the face

of increasing competition. A complete analysis would require a vast range of application types to be tested, however it is clear that decisions to set up a business in a specific location and decisions about configurations of the services on which the business depend strongly on the business type, the business growth stage and the characteristics of service quality in the markets within which they compete.

Acknowledgments

This work was funded under the Wellcome Trust, Grant WT086696MA, a HECToR Distributed Computational Science and Engineering (dCSE) support award administered by NAG Ltd, ESRC 'Follow-On Funding' award RES-189-25-0039 for the INWA project, the UK EPSRC award EP/H006753/1 on Building Relationships with the 'Invisible' in the Digital (Global) Economy and finally on UK EPSRC award EP/H043160/1 SSI: The UK Software Sustainability Institute. The authors gratefully acknowledge the assistance of Dr. Sornthep Vannarat, Head of the Large Scale Simulation Laboratory from the National Electronics and Computer Technology Center when performing the experiments in Thailand.

References

1. Heller MJ. DNA microarray technology: devices, systems, and applications. Annual review of biomedical engineering. 2002; 4:129–153.
2. Xu L, German D, Winslow RL. Large-scale integration of cancer microarray data identifies a robust common cancer signature. BMC Bioinformatics. 2007; 8:1471–2105.
3. Edgar R, Domrachev M, Lash AE. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. Nucleic Acids Res. Jan 1; 2002 30(1):207–10. [PubMed: 11752295]
4. Parkinson, et al. ArrayExpress update—an archive of microarray and high-throughput sequencing-based functional genomics experiments. Nucl. Acids Res. doi: 10.1093/nar/gkq1040. Pubmed ID 21071405.
5. Shendure J, Ji H. Next-generation DNA sequencing. Nature biotechnology. 2008; 26:1135–1145.
6. Richter BG, Sexton DP. Managing and analyzing next-generation sequence data. PLoS Computational Biology. 2009; 5:e1000369. [PubMed: 19557125]
7. Gentleman, R.; Carey, V.; Huber, W.; Irizarry, R.; Dudoit, S., editors. Bioinformatics and Computational Biology Solutions Using R and Bioconductor. Springer; 2005.
8. The R Project for Statistical Computing. http://www.r-project.org/
9. Hill J, Hambley M, Forster T, Mewissen M, Sloan TM, Scharinger F, Trew A, Ghazal P. SPRINT: a new parallel framework for R. BMC Bioinformatics. 2008; 9:558. [PubMed: 19114001]
10. Petrou, S.; Sloan, TM.; Mewissen, M.; Forster, T.; Piotrowski, M.; Dobrzelecki, B.; Ghazal, P.; Trew, A.; Hill, J. Optimization of a parallel permutation testing function for the SPRINT R package Concurrency and Computation: Practice and Experience. 2011. http://dx.doi.org/10.1002/cpe.1787
11. Piotrowski, M.; Forster, T.; Dobrezelecki, B.; Sloan, TM.; Mitchell, L.; Ghazal, P.; Mewsissen, M.; Petrou, S.; Trew, A.; Hill, J. Optimisation and parallelisation of the partitioning around medoids function in R. International Conference on High Performance Computing and Simulation (HPCS); 2011. p. 707-713.
12. Mitchell, L.; Sloan, TM.; Mewissen, M.; Ghazal, P.; Forster, T.; Piotrowski, M.; Trew, AS. A parallel random forest classifier for R. Proceedings of the second international workshop on Emerging computational methods for the life sciences ECMLS '11; 2011. p. 1-6.
13. Amazon, Elastic Cloud (EC2). [Accessed 27th October 2011] http://aws.amazon.com/ec2/
14. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. Journal of Molecular Biology. 1990; 215:403–410. [PubMed: 2231712]
15. Matsunaga, A.; Tsugawa, Ma; Fortes, J. CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications. IEEE Fourth International Conference on eScience; 2008. p. 222-229.

16. Lu, W.; Jackson, J.; Barga, R. AzureBlast : A Case Study of Developing Science Applications on the Cloud. HPDC '10 Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing; 2010. p. 413-420.
17. McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, DePristo M. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome research. 2010; 20:1297–1303. [PubMed: 20644199]
18. Schmidberger M, Morgan M, Eddelbuettel D, Yu H, Tierney L, Mansmann U. State of the Art in Parallel Computing with R. Journal of Statistical Software. Aug; 2009 Vol. 31(No. 1):1–27.
19. Yu H. Rmpi: Parallel Statistical Computing in R. RNews. 2002; 2(2):10–14.
20. [Accessed 27th Oct 2011] Amazon EC2 FAQs. http://aws.amazon.com/ec2/faqs/
21. Clarke, L.; Glendinning, I.; Hempel, R. The MPI Message Passing Interface Standard. Programming Environments for massively distributed systems: working conference of the IFIP WG10.3; Monte Verita, Switzerland. 1994.
22. Dobrzelecki, B.; Krause, A.; Piotrowski, M.; Hong, N Chue. Grid and Cloud Database Management. Springer; Berlin Heidelberg: 2011. Managing and Analysing Genomic Data Using HPC and Clouds.
23. [Accessed 31/10/2011] MPICH2. http://www.mcs.anl.gov/research/projects/mpich2/
24. Nadgowda, SJ.; Sion, R. Cloud Performance Benchmark Series: Amazon Elastic Block Store (EBS); Amazon Simple Storage Service (S3; Amazon EC2 Instance Local Storage v0.8. Cloud Computing Center, Stony Brook Network Security and Applied Cryptography Lab;
25. Walker E. Benchmarking Amazon EC2 for high-performance scientific computing. Login. 2008; 33:18–23.
26. He, Q.; Shujia, Z.; Kobler, B.; Duffy, D.; McGlynn, T. Case study for running HPC applications in public clouds. HPDC'10 Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing; 2010. p. 395-401.
27. Hammond, M.; Hawtin, R.; Gillam, L.; Oppenheim, C. Cloud computing for research. Curtis and Cartwright; Jun. 2010 Technical Report
28. Iakymchuk, R.; Napper, J.; Bientinesi, P. Improving high-performance computations on clouds through resource underutilization. Proceedings of the 2011 ACM Symposium on Applied Computing; 2011. p. 119-126.
29. Iosup, A.; Yigitbasi, N.; Epema, D. On the performance variability of production cloud services. Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing; 2011.
30. Schad J, Dittrich J, uian e-Ruiz J. Runtime measurements in the cloud: observing, analyzing, and reducing variance. Proc. VLDB Endow. Sep.2010 3:460–471.
31. Hill, Z.; Humphrey, M. A quantitative analysis of high performance computing with amazon's ec2 infrastructure: The death of the local cluster?. 10th IEEE/ACM International Conference on Grid Computing; 2009. p. 26-33.
32. Jackson, KR.; Ramakrishan, L.; Muriki, K.; Canon, S.; Cholia, S.; Shalf, J.; Wasserman, HJ.; Wright, NJ. Performance analysis of high performance computing applications on the Amazon Web Services cloud. Proceedings of IEEE Second International Conference on Cloud Computing Technology and Science; 2010.
33. Armbrust, M.; Fox, A.; Griffith, R.; Joseph, AD.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; Zaharia, M. Above the clouds: A Berkeley view of cloud computing. UC Berkeley Reliable Adaptive Distributed Systems Laboratory; Feb. 2009 Technical Report
34. [Accessed 31/10/2011] Amazon Web Services High Performance Computing. http://aws.amazon.com/hpc-applications/
35. Petrou, S. [Accessed 09/05/2011] SPRINTing with HECToR. 2010. http://www.hector.ac.uk/cse/distributedcse/reports/sprint/sprint.pdf
36. Oosterhuis, F.; Dodoková, A.; Gerdes, H.; Greño, P.; Jantzen, J.; Mudgal, S.; Neubauer, A.; Rayment, M.; Stocker, A.; Tinetti, B.; van der Woerd Varma, A. The use of differential VAT rates to promote changes in consumption and innovation, Final Report under DG Environment. 2008. Contract 070307/2007/482673/G1

37. [Accessed 17/03/2011] This data is available from the Documents section of http://www.r-sprint.org/

38. Quackenbush J. Computational Approaches to Analysis of DNA Microarray Data. Methods of Information in Medicine. IMIA Yearbook 2006: Assessing Information - Technologies for Health. 2006; 1:91–103.

39. Bolshakova N, Azuaje F. Computational Approaches to Analysis of DNA Microarray Data. Estimating the Number of Clusters in DNA Microarray Data. Methods of Information in Medicine. 2006; 45(2):153–157. [PubMed: 16538280]

Europe PMC Funders Author Manuscripts

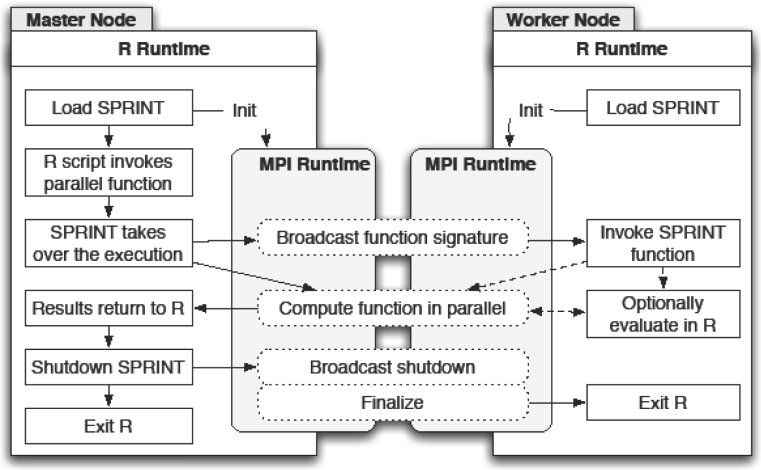

Figure 1.
The SPRINT Framework Architecture as described in [22].

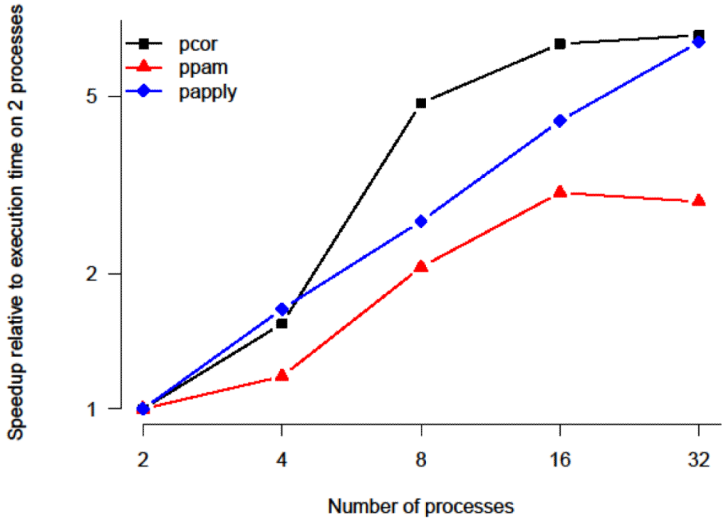

Figure 2.
Speed-up of pcor, ppam, papply in the workflow for a dataset with 14,821 genes and 75 biological samples

Europe PMC Funders Author Manuscripts

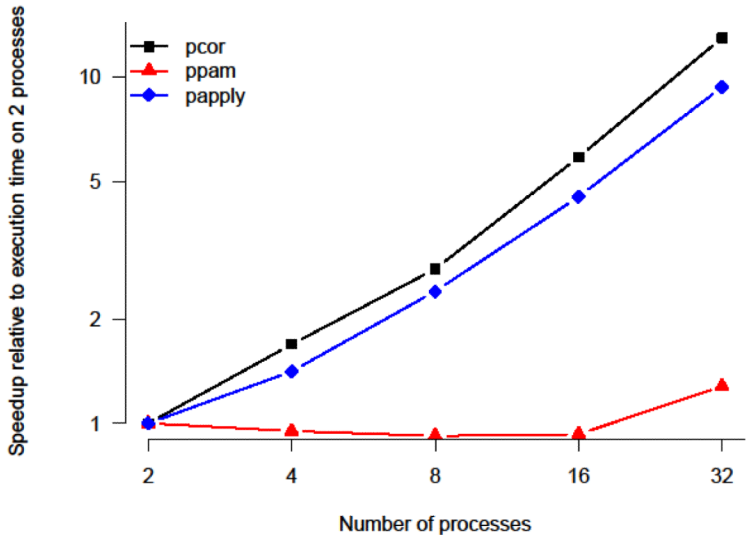

Figure 3.
Speed-up for pcor, ppam and papply in the workflow on a dataset of 29,642 genes and 75 biological samples

Europe PMC Funders Author Manuscripts

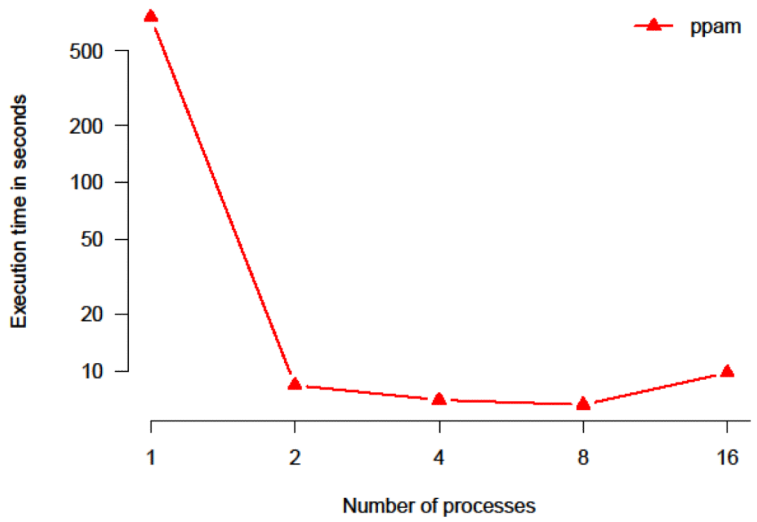

Figure 4.
Execution time of the serial R partitioning around medoids function on 1 processor versus its SPRINT ppam optimised version on multiple processors (2, 4, 8, 16).

Europe PMC Funders Author Manuscripts

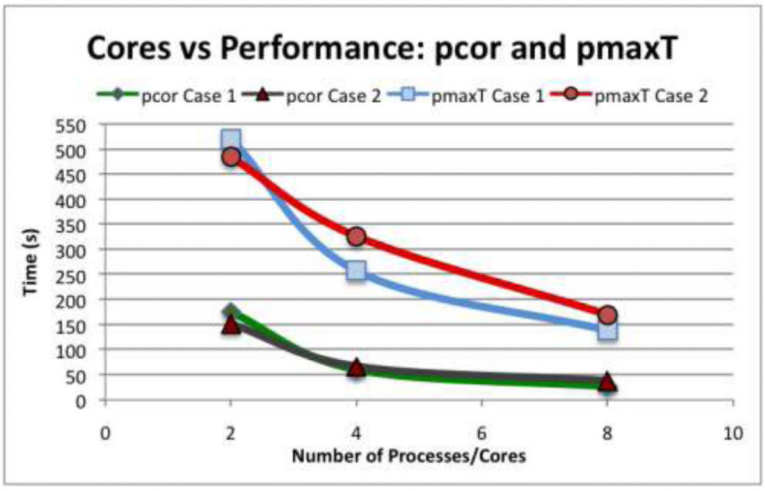

Figure 5.
Time taken to complete runs for pcor and pmaxT in different resource utilization scenarios

Europe PMC Funders Author Manuscripts

Table I

Difference in Resource Usage between the UK and Thailand

Location	Cost	IDT Data In	IDT Data Out	Storage	I/O Req.
UK	$2.52	0.274 GB	0.008 GB	0.151 GB	46,523
Thailand	$2.10	0.205 GB	0.007 GB	0.151 GB	84,103