Economics of Cloud Computing: a Statistical Genetics Case Study

Jeremy M. R. MARTIN

Steven J. BARRETT

Simon J. THORNBER

Silviu-Alin BACANU

Dale DUNLAP (UnivaUD)

Steve WESTON (Revolution Computing)

3rd November 2009

Some Definitions

- *Grid Computing:* combining multiple computer resources to solve a single task, typically a scientific, technical or business problem that requires a great number of processing cycles or the need to process large amounts of data.

- *Cloud Computing:* a paradigm of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet.

- *Genetic Association Analysis:* statistical analysis of data from many patients to link a disease to a genetic mutation, potentially leading to discovery of new medicines.

- *R:* a very powerful and high level programming language for statistical analysis and data visualisation.

Project Summary

Aim: Investigate the *feasibility* and *economics* of running a major genetic association analysis on external 'clouds'.

Method: Set up three way collaboration between GSK, Revolution Computing (statistical software specialists) and Univa (Cloud brokers) to run a Cloud Computing PoC using our parallel R software

Three strategies for cost reduction:

1. Make efficient use of external resources with Univa's scheduler and resource manager – keep the rented cloud resources busy

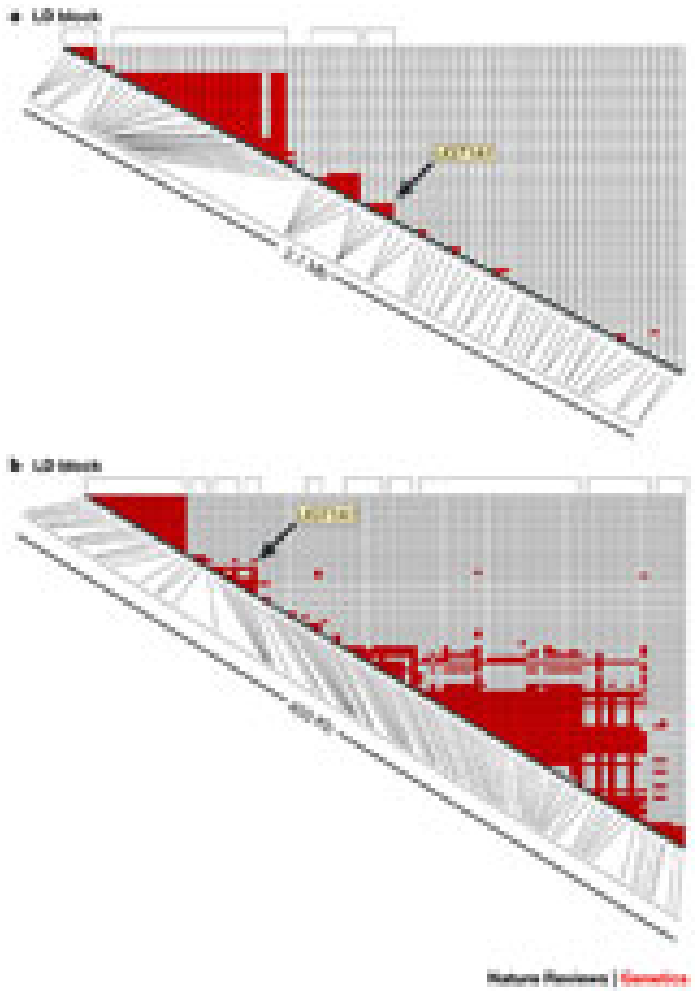
2. Optimise the serial performance of the R code – using Revolution's expertise and toolset

3. Seek out the lowest cost cloud computing resources to run the application.

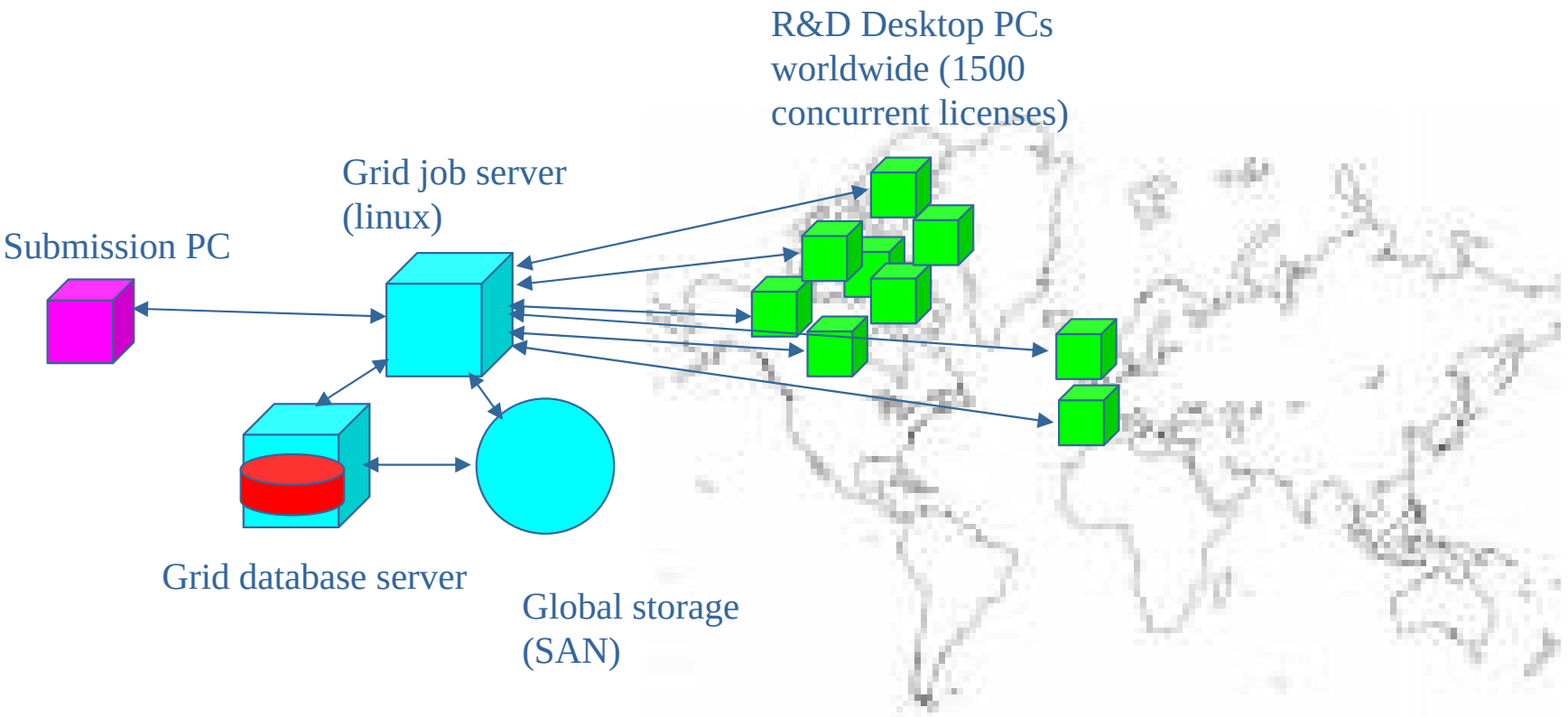
Genetic Association Analysis using Simulation
Silviu Alin Bacanu

SNP association analysis:

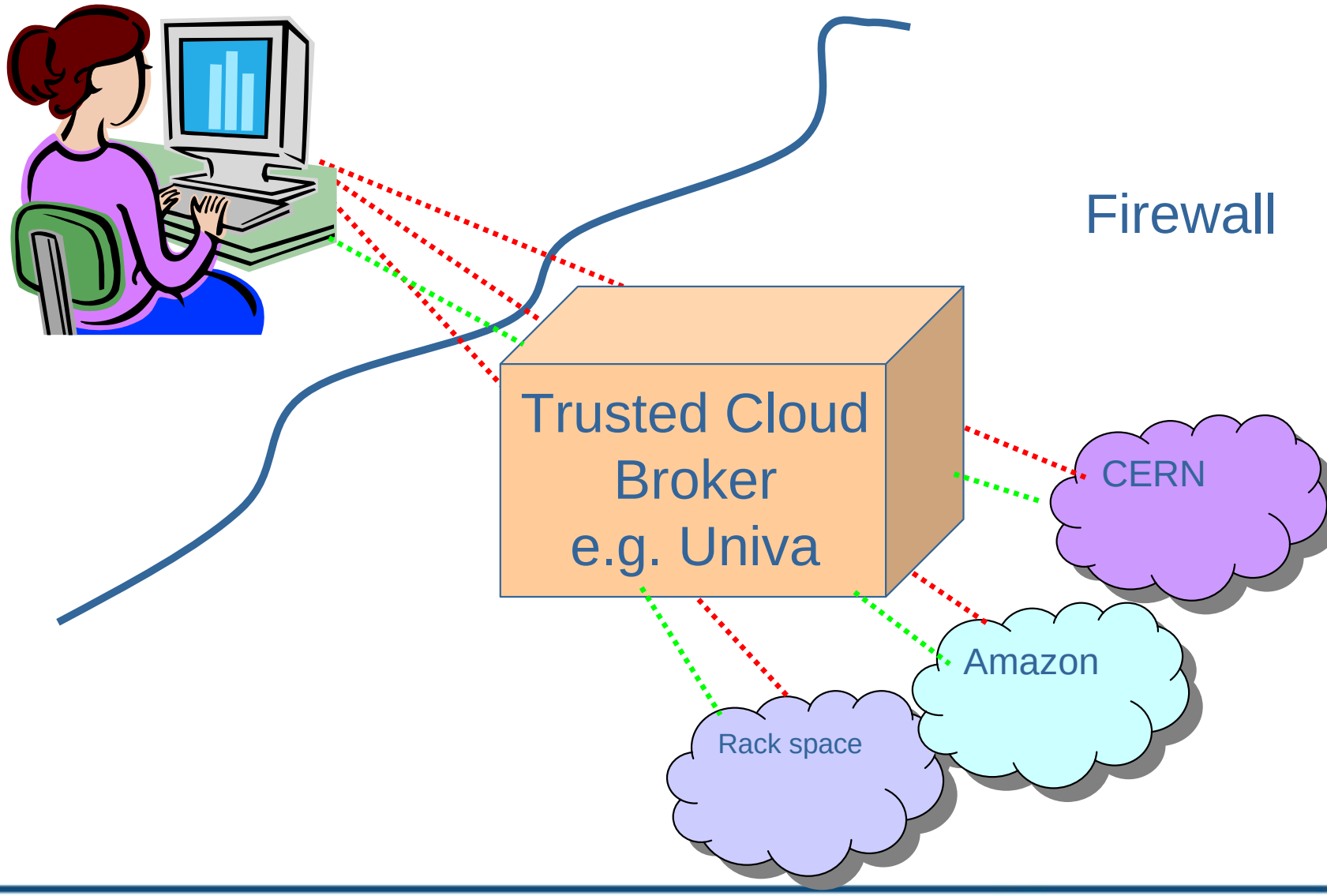
- 5000 cases, 5000 controls
- *Phenotype* and *genotype* data
- Run 250,000 sub-analyses each with a different combination of parameters and 1000 different permutations of the data.
- Total CPU time: 7 years
- Elapsed time on GSK desktop grid: ~3 days



GSK Desktop Grid



Cloud Computing



Obstacles to Cloud Computing.

Source: **Above the Clouds: A Berkeley View of Cloud Computing** Technical Report No. UCB/EECS-2009-28 http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

1 Availability of Service

2 Data Lock-In

3 Data Confidentiality and Auditability

4 Data Transfer Bottlenecks

5 Performance Unpredictability

6 Scalable Storage

7 Bugs in Large Distributed Systems

8 Scaling Quickly

9 Reputation Fate Sharing

10 Software Licensing

Strategy 1. Efficient Resource Management

UniCloud works as follows:

- One Linux virtual machine (“installer node”) is created within the chosen cloud environment (e.g. Amazon EC2).
- Additional virtual machines are created programmatically using the vendor cloud API.
- Sun Grid Engine is installed by Unicloud and used as the batch job scheduler.
- Jobs are queued by SGE and then farmed out to different machines as they become available.

Strategy 2: Accelerating R Code

- R programming language is a popular and productive public-domain tool for statistical computing and graphics.
- However R programs may take a long time to execute when compared with equivalent programs written in low-level languages like C so there have been many initiatives to make R programs run faster.
- These fall into three general categories:
  - Task farm parallelisation: running a single program many times in parallel with different data across a grid or cluster of computers.
  - Explicit parallelisation in the R code using MPI or parallel loop constructs (e.g. R/Parallel).
  - Speeding up the performance of particular functions by improved memory handling, or by using multithreaded or parallelised algorithms 'beneath the hood' to accelerate particular R functions e.g. REvolution R, Parallel R or SPRINT.
- We are using a combination of approach 1 with approach 3.

Results of Serial Optimisation of the R code

- Using the public domain Revolution version of R improved the execution time by 6%.
- Code profiling using the RProf tool revealed that most of the execution time was within the ‘lm’ function (and related functions) for fitting linear models, based on the QR matrix factorization.
- This has not yet been optimised by Revolution Computing but is work in progress and should provide a further 5-10% improvement.
- A simple code transformation provided a further 20% improvement: transforming a ‘for’ loop iteration to a function together with an ‘sapply’ command.

Strategy 3: Pursuit of Low Cost Compute Cycles Amazon Elastic Cloud (EC2) Linux Instances

Instance	Memory GB	'Compute units'	Storage GB	Architecture	Price per VM hour
Standard small	1.7	1	160	32 bit	$0.10
Standard large	7.5	4	850	64 bit	$0.20
Standard extra large	15	8	1690	32 bit	$0.40
High CPU medium	1.7	5	350	32 bit	$0.20
High CPU large	7	20	1690	64 bit	$0.80

Rackspace Linux Instances

Memory MB	Storage GB	Price per VM hour
256	10	$0.015
512	20	$0.03
1024	40	$0.06
2048	80	$0.12

Association analysis code resource requirements:

- 50MB RAM
- Negligible storage
- 15 minutes CPU time on modern Intel processor
- 250,000 times over

Final Results

Cloud virtual machine	*Cost/hr*	*Throughput: jobs/inst/hr*	*Estimated total cost of run*
Amazon EC2 Standard XL (8 EC2 compute units, 15GB RAM)	$0.80	8.25	$24,250.00
Amazon EC2 Standard XL following R code optimisation	$0.80	12.5	$16,000.00
Amazon EC2 High CPU XL (20 EC2 compute units, 7 GB RAM)	$0.80	26.82	$7,458.33
Amazon EC2 High CPU XL following R code optimisation	$0.80	35.56	$5,625.00
Rackspace 256MB RAM	$0.015	12.23	$306.72
Rackspace 256MB RAM, following R code optimisation	$0.015	15.24	$246.09

Caveats

- This is a best case scenario for Rackspace – performance could be adversely affected by other users on the system.
- These results are specific to a particular class of problem. Applications requiring huge amounts of data might not map so effectively to cloud computing.
- Usage of Rackspace is limited to 200 concurrent VMs, which gives a best execution time of 4 days for this problem. Amazon can scale much higher.
- There is no cost from cloud broker included here – just the cost of the cloud cycles.

Conclusions

- Going forward we foresee a model of cloud brokerage emerging whereby a layer of middleware is provided to help satisfy customers constraints in utilising software services based on factors such as cost, security or overall execution time.

- One possible attractive feature would be for the cloud broker to charge the customer a fixed cost for *total compute cycles*, rather than virtual CPU time.
  - The Broker would then be taking on the risk of performance degradation on a third party cloud.
  - They would need to build performance monitoring into their resource manager.

- If you can find a vendor that has an instance that perfectly matches your workload (or, if it's possible to vary your workload to perfectly match an inexpensive instance), cloud computing can be *very* inexpensive.