Genome Analysis in a Dynamically Scaled Hybrid Cloud

Chris Smowton*, Georgiana Copil**, Hong-Linh Truong**, Crispin Miller* and Wei Xing*
* CRUK Manchester
** TU Vienna
In a Nutshell

- Users want to run analysis tasks
- Admin wants to dynamically allocate resources
- Non-expert users are bad at assigning resources to their jobs
- We propose to automatically control global and per-job parallelism...
- ...and show via simulation that this could work
Background: CELAR Project

• Cloud applications are often scalable
• Example: web database frontend
  – How many database servers?
  – How many web servers?
  – What spec cloud instance for each one?
• Parameters adjusted in ad-hoc manner, using admin’s intuition
• CELAR: automate setting these parameters instead
Background: CELAR Project

• Applications expose *metrics*
  – Web example: avg. page load time, web server memory usage, ...

• Admin sets *goals*
  – Keep page load time below 1 second

• CELAR can hire and release cloud resources

• Learns how different resources help achieve the goals
Background: Resource Managers

• Users submit jobs
  – Annotated with *resource needs*

• Admin provides machines (usually fixed physical hardware, maybe cloud-hired)

• Resource manager (RM) queues jobs and assigns jobs to machines
CELAR-ised Resource Manager

• Expose metrics to CELAR:
  – Job queue length, user “happiness”

• Set goals:
  – Maximise happiness!

• Let CELAR pick how many cloud VMs the RM gets, and what spec they have
This Work: Show that CELARising our RM can be a good thing

• CELAR chooses how many VMs the RM has at its disposal, and their spec
• Show that there is scope to improve over simple control policies
  – For example, hire a VM whenever there is a queue
• Evaluate by simulating interaction between users, resource manager and CELAR
Model

Users
Submit jobs (randomly)

CELAR
Observes metrics; takes hiring decisions

Resource Manager
Provides variable cost machines (2-tier)

Infrastructure
Model Users

• Users pay for service; we pay the infrastructure provider to hire machines
  – In practical academic environments, instead of money these might be *quota credits*
• Users pay more for “better” service
• We try to make as much “profit” as possible
Model Application: GATK

• 7-stage analysis pipeline
• Users care how long the whole pipeline takes
• Different stages have different scaling behaviour
  – Measured through profiling
CELAR Task 1: When to hire, when to wait?

• Infrastructure has two cost tiers: cheap and expensive
• Suppose a queue is building up at the RM...
• ...but no cheap resources are available
• Should we hire an expensive machine or wait for a cheap one?
Proposed solution: Predict

- From our GATK profiling we can predict when running tasks will finish
- Weigh amount the user will pay for faster completion vs. expected extra cost
- If the queue is long, *all* queued jobs will benefit if we accelerate one
Predictive Scaling Example

Profit vs. mean arrival interval for various horizontal scaling functions

Reward function: Time-based
Public-tier hire cost (CUs per TU): 80
Resource allocation algorithm: Best constant plan

Horizontal scaling function:
- Predictive
- Always-scale
- Never-scale
Caveat: Prediction Quality

Profit vs. mean arrival interval for various error factor standard deviations

Horizontal scaling function: Predictive
Reward function: Time-based
Resource allocation algorithm: Greedy

Error factor standard deviation
- 0
- 0.1
- 0.2
- 0.3
CELAR Task 2: To Thread or Not To Thread?

• Most GATK pipeline stages support multithreading...

• ...but some better than others, leading to diminishing returns

• During idle times, best to run multithreaded and make the users happy

• During busy times, best to run singlethreaded and maximise throughput
Proposed Solution: Weigh Expected Reward vs. Extra Cost

• Profiling lets us predict the effect of adding a thread
• Compare against cost, taking into account whether we would use cheap or expensive resources
• Three candidate schemes: greedy, long-term and long-term adaptive
Vertical Scaling Example

Profit vs. mean arrival interval for various resource allocation algorithms

- Horizontal scaling function: Always-scale
- Reward function: Throughput-based
- Public-tier hire cost (CUs per TU): 50

Mean profit per pipeline run (CUs)

Mean inter-arrival interval (TUs)
Caveat: Greed?

• If there are only 8 cheap cores left, and the next job would ideally use them all, should we really grab them all ourself?
• Surprisingly, yes!
Summary

• Developed candidate algorithms for automatically controlling the size of an RM’s pool of worker VMs and the parallelism used by each job

• Showed ability to improve over baseline algorithms
What’s Next?

• Apply findings in the real world: use auto-scaling on our real cluster
• Compare against existing work on e.g. Hadoop scheduling, where jobs are flexible by default
• Demonstrate within the CELAR framework
More Information

www.celarcloud.eu
christopher.smowton@cruk.manchester.ac.uk