Improving HPC applications scheduling with predictions based on automatically-collected historical data

Carlos Fenoy García
carles.fenoy@bsc.es

September 2014

Index

Introduction

- Supercomputers have increased their resource number in last years

$$\Downarrow$$

Resource management has become critical
to maximise its usage due to resource sharing

- The evolution pace of different components is not equal on all the parts: Memory bandwidth evolves slower than processors speed.
- Existing resource selection mechanisms only focus on CPUS and Memory. Other limiting resources exist
  - Interconnect bandwidth
  - Memory bandwidth

Introduction

- Previous work exist to solve the memory bandwidth management issue. The PhD thesis by Francesc Guim introduces the Less Consume selection policy, that considers the memory bandwidth as a new resource.
- Less Consume was later validated, porting the policy to a real system and analysing its behaviour.
- However, this port was done on MareNostrum 2, a different arquitecture to the one currently available in MareNostrum 3.
- This policy requires users to specify the amount of memory bandwidth needed by each job.

Motivation

- Users do not necessarily know the resources used by their applications.
- The impact of sharing resources on the applications perfomance.
- Improvement of monitoring systems enables the collection of multiple metrics per job.
- The lack of trusty mechanisms for specifying resources.

Objectives

- Port previous work to a new arquitecture (MareNostrum 3).
- Enhance an existing resource selection policy with the usage of historical data to predict the resource usage by jobs.
  - Aware of shared resources (memory bandwidth)
  - Historical data collected by a transparent monitoring system

$\Downarrow$

Data is used in job scheduling

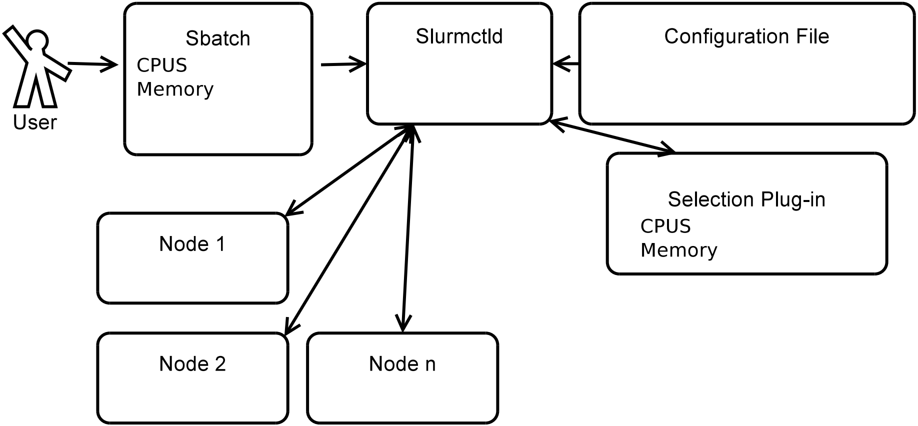
- Analyse the behaviour and the benefits of the policy
  - Compare the policy with other existing policies

Index

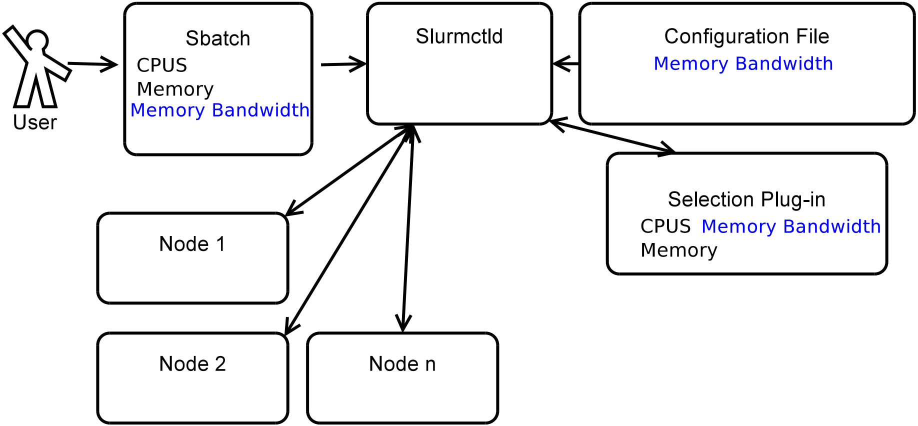
Historical Performance Data (HPerf)

- The system proposed predicts the resources usage of an application based on monitoring information gathered from previous equivalent executions.
- This system uses a user-provided tag to identify the kind of job.
- Combines existing technologies to improve the scheduling of applications:
  - Resource selection policy aware of memory bandwidth (Less Consume)
  - Monitoring system able to collect per job information (PerfMiner)
  - Scheduler and resource management (Slurm)

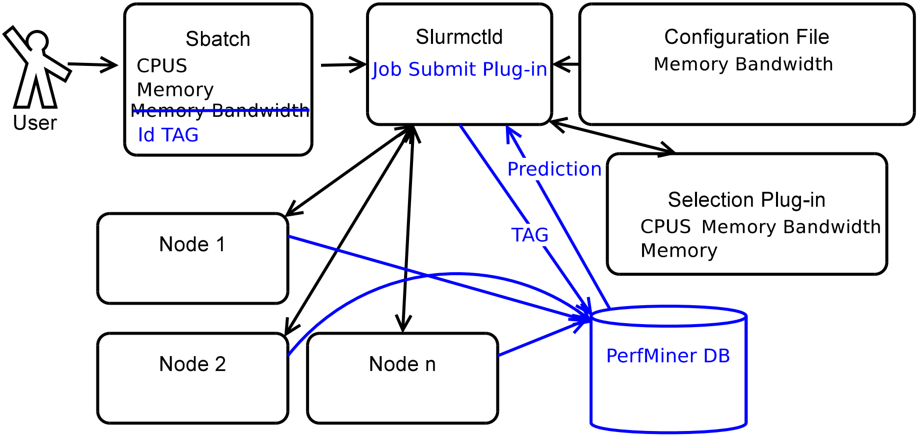
Consumable Resources (ConsRes)



Less Consume (LessC)



Historical Performance data (HPerf)



- If at the submission time the tag is found, the average of the resource usage will be used as the resource requirements for the job.
- If it is not found, the application will run with exclusive execution to favour the monitoring.

Index

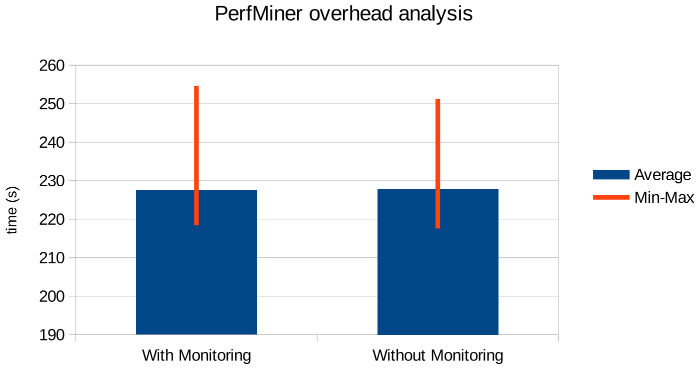
System analysis

In order be able to analyse the system, the following previous work was done:

- PerfMiner overhead analysis.
- Applications characterization.
  - Study of the resources used by an application.
  - Study of the time elapsed per iteration for each application.
- Workload generation:
  A workload generator was chosen due to the unfeasibility of using real production applications.

PerfMiner study

Overhead analysis of PerfMiner with two sets of 100 jobs.
(CG class D with 64 tasks)



Workload generation: applications characterization

Applications used for the generation of the workload characterized by their use of memory bandwidth:

- High
  - CG class D
  - Synthetic application with high memory bandwidth usage
- Medium
  - CG class C
- Low
  - CG class B

Workload generation

The Lublin-Feitelson model was used to generate the workload.

- Generates 100 jobs
- Provides:
  - Number of cpus (2-64)
  - Job duration
  - Job arrival time

Combining the workload with the applications list 2 final workloads were obtained:

	Medium	High
high	54	84
medium	27	7
low	19	9

The time limit used for the jobs of the workload was the calculated with the application running standalone.

Index

Description

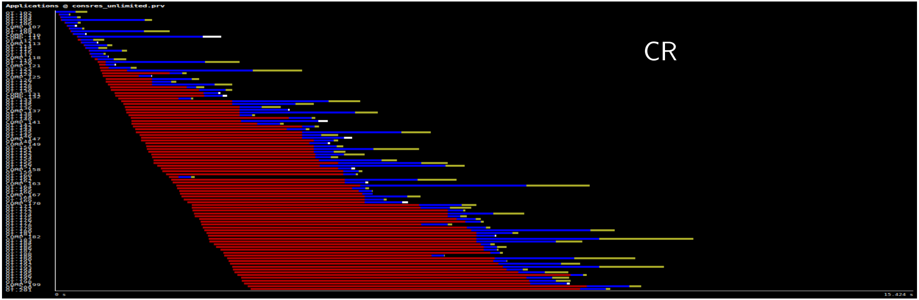
Both workloads (HIGH & MEDIUM) were run with 3 different scheduling policies:

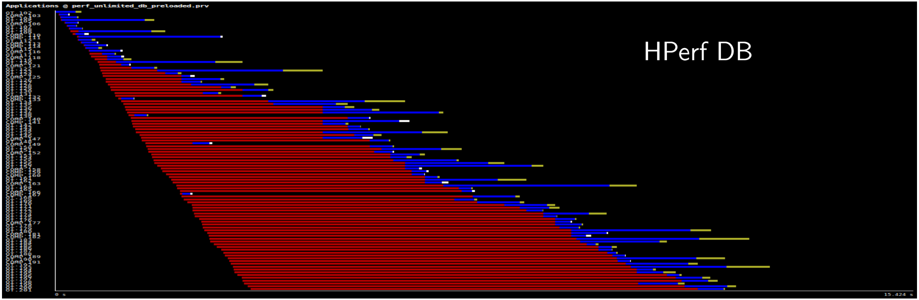
- ConsRes: Default Slurm scheduling policy. Considers only cpus and memory.
- LessC: Less Consume policy aware of the memory bandwidth usage per application.
- HPerf: Less Consume + Historical Perfomance data. Automatically gets the jobs resource consumption.
  - With empty database.
  - With preloaded database.

For each policy tests were run with:

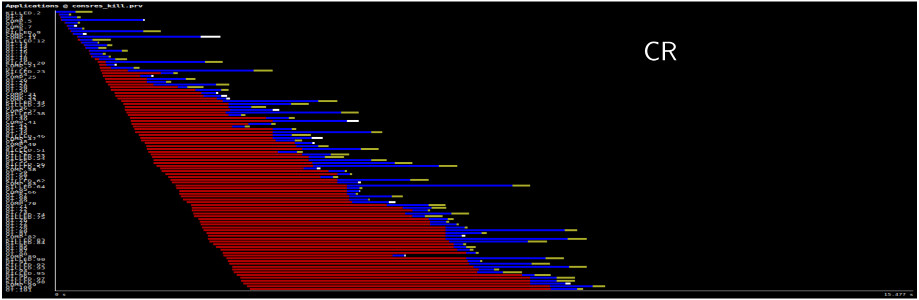
- Unlimited grace time.
- Limited grace time. Kills the jobs after 5 minutes of the time limit.

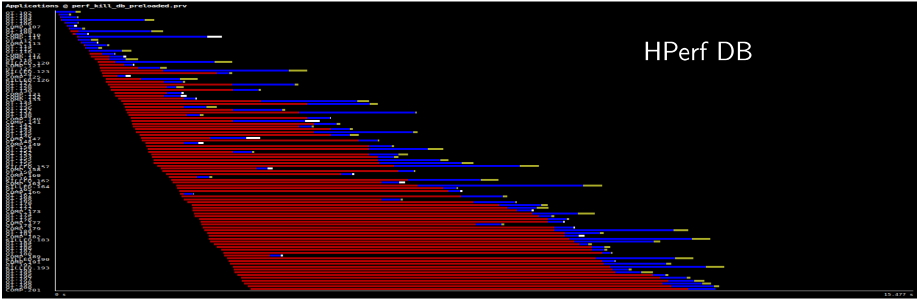
HIGH: Unlimited time





HIGH: Limited grace time

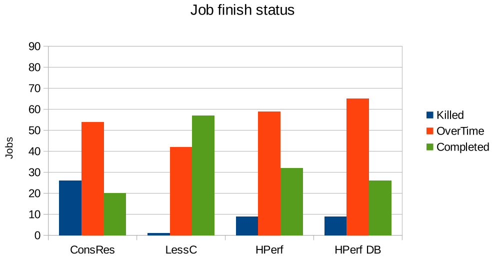




Finishing status

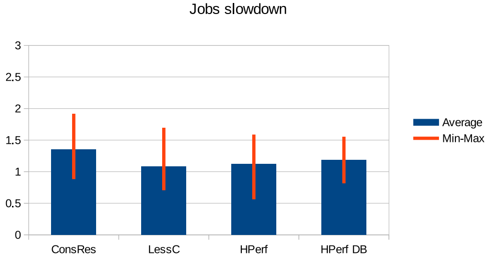
Unlimited

Limited

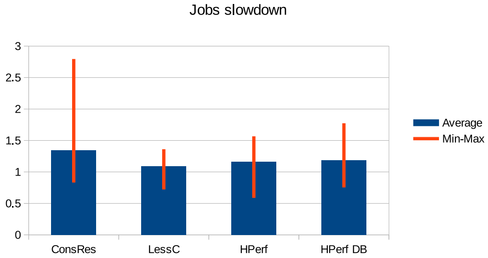


Slowdown

Unlimited

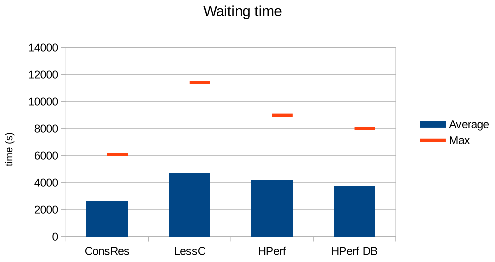


Limited

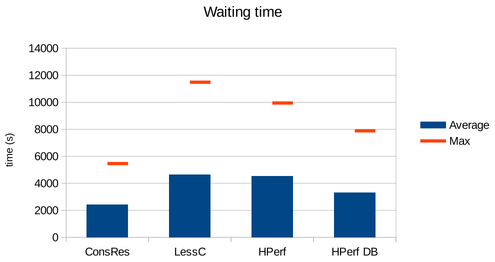


Waiting time

Unlimited

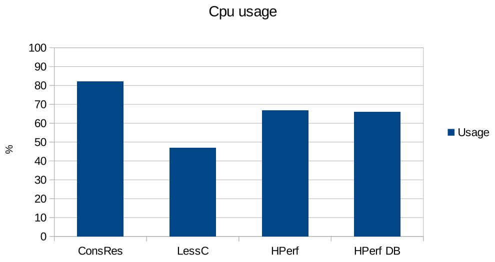


Limited

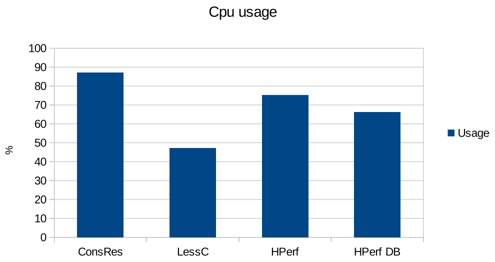


CPU usage

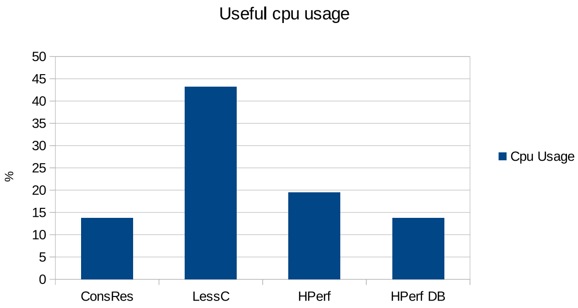
Unlimited



Limited

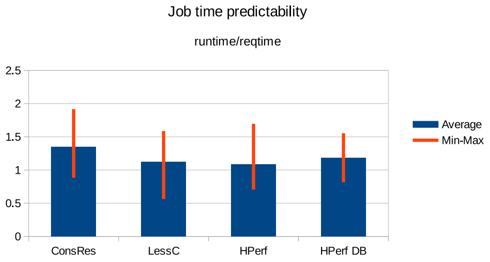


Useful CPU usage

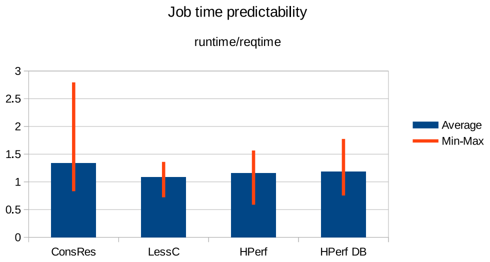


Job run time predictability

Unlimited



Limited



Index

Conclusions

- The usage of the monitoring data for job scheduling results in better allocation that:
  - Improves the applications performance.
  - Increases the useful cpu usage.
  - Reduces the shared resources overload.
  - Increases the waiting time.
- Avoids users providing information they may not know.
- Requires better mechanisms to measure the memory bandwidth to increase the solution performance.

Improving HPC applications scheduling with predictions based on automatically-collected historical data

Carlos Fenoy García
carles.fenoy@bsc.es

September 2014