Dynamic Resource Provisioning in Grid Environments

Ioan Raicu
Computer Science Department
The University of Chicago
iraicu@cs.uchicago.edu

Catalin Dumitrescu
Computer Science Department
The University of Chicago
catalind@cs.uchicago.edu

Ian Foster
Math and Computer Science Div., Argonne National Lab.
Computer Science Dept., The University of Chicago
foster@mcs.anl.gov

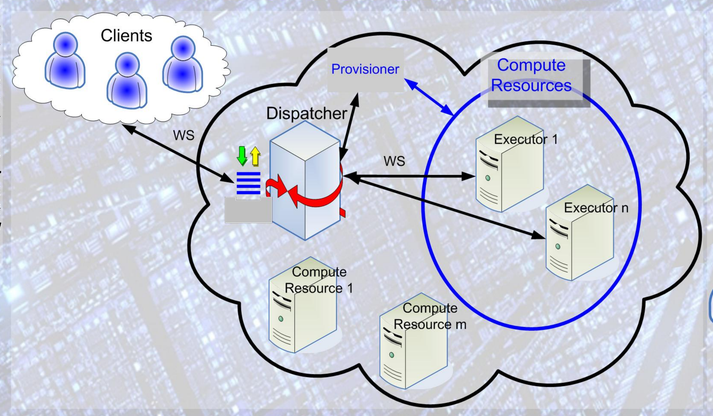
Abstract

Batch schedulers commonly used to manage access to parallel computing clusters are not typically configured to enable easy configuration of application-specific scheduling policies. In addition, their sophisticated scheduling algorithms can be relatively expensive to execute. Thus, for example, applications that require the rapid execution of many small tasks often do not perform well. It has been proposed that these problems be overcome by separating the two tasks of provisioning and scheduling. This paper focuses on resource provisioning, the various allocation and de-allocation policies, and how dynamic and adaptive provisioning can be in light of varying workloads. We couple the proposed dynamic resource provisioning (DRP) with an existing system, Falkon, which is used for the scheduling of tasks to the provisioned resources. We describe the DRP architecture and implementation, and present performance results for both microbenchmarks and applications. Microbenchmarks show that DRP can allocate resources on the order of 10s of seconds across multiple Grid sites and can reduce average queue wait times by up to 95% (effectively yielding queue wait times within 3% of ideal); furthermore, applications (executed by the Swift parallel programming system) reduce end-to-end run time of up to 90% for large-scale astronomy and medical applications, relative to versions that execute tasks via separate scheduler submissions.

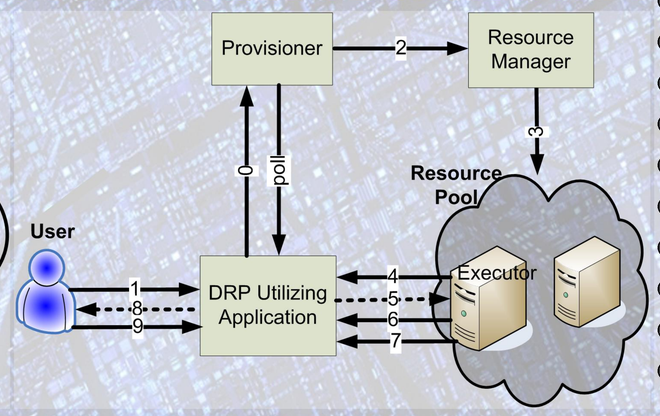
Goals

- **Use an adaptive provisioner to acquire and/or release resources as application demand varies**
- **Reduce average queue wait times by amortizing high overhead of resource allocation over the execution of many tasks**

Falkon Architecture



Provisioner Architecture

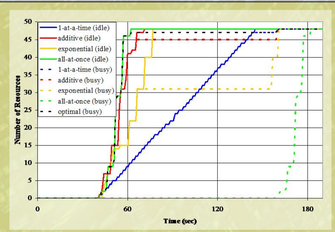


(0) DRP Utilizing Application ==> Provisioner: **Initialization call**
(1) User ==> DRP Utilizing Application: **Task submission**
(2) Provisioner ==> Resource Manager: **Resource allocation request**
(3) Resource Manager ==> Local Resource Manager: **Resource allocation and boot-strapping**
(4) Executor ==> DRP Utilizing Application: **Executor registration**
(5) DRP Utilizing Application ==> Executor: **Available task notification**
(6) Executor ==> DRP Utilizing Application: **Task pickup**
(7) Executor ==> DRP Utilizing Application: **Task results delivery**
(8) DRP Utilizing Application ==> User: **Task completion notification**
(9) User ==> DRP Utilizing Application: **Task results pickup**

Performance Results

Acquisition Policy

1. **One-at-a-time:** for n resources, n allocations are made
2. **Additive:** for n resources, $\lceil (\sqrt{8n+1}-1)/2 \rceil$ allocations are made
3. **Exponential:** for n resources, $\lceil \log_2(n+1) \rceil$ allocations are made
4. **All-at-once:** for n resources, 1 allocations are made
5. **Optimal:** gets the most available resources in single allocation requests



Strategy	Accumulated CPU Time IDLE	Accumulated CPU Time BUSY
1-at-a-time	4220 sec	4205 sec
additive	6048 sec	5773 sec
exponential	5702 sec	4267 sec
all-at-once	6156 sec	409 sec
optimal	6059 sec	6059 sec

Release Policy

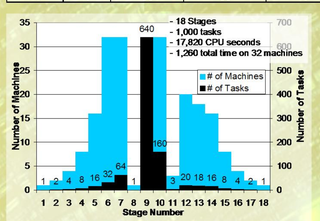
Stage #	# of Tasks	Task Time (sec)	Stage Time (32 machines)	# Machines (32 max)
1	1	60	60	1
2	2	60	60	2
3	4	60	60	4
4	8	60	60	8
5	16	60	60	16
6	32	60	60	32
7	64	60	120	32
8	1	120	120	1
9	640	6	120	32
10	160	12	60	32
11	3	60	60	3
12	20	60	60	20
13	18	60	60	18
14	16	60	60	16
15	8	60	60	8
16	4	60	60	4
17	2	60	60	2
18	1	60	60	1

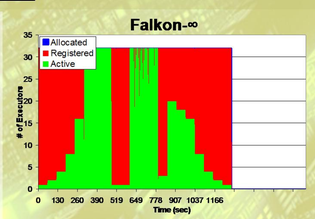
- **Compared:**
  - **GRAM+PBS:** 1 job per task
  - **Falkon-15, -60, -120, -180:** dynamic resource provisioning – 15, 60, 120, and 180 seconds idle time before resource de-allocation, and with 32 maximum machines
  - **Falkon-∞:** static resource provisioning with 32 maximum machines
  - **Ideal** with 32 maximum machines
- **Allocated** (blue): LRM wait queue
- **Registered** (red): idle executors
- **Active** (green): executors processing tasks

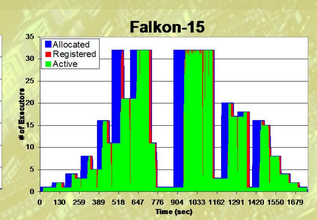
- **End-to-end execution time:**
  - 1260 sec in ideal case
  - 4904 sec ==> 1276 sec
- **Average task queue time:**
  - 42.2 sec in ideal case
  - 611 sec ==> 43.5 sec
- **Trade-off:**
  - Resource Utilization for Execution Efficiency

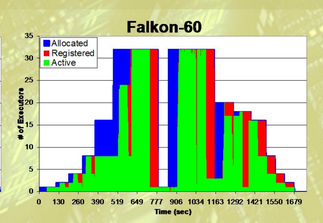
	GRAM +PBS	Falkon-15	Falkon-60	Falkon-120	Falkon-180	Falkon-∞	Ideal (32 nodes)
Queue Time (sec)	611.1	87.3	83.9	74.7	44.4	43.5	42.2
Execution Time (sec)	56.5	17.9	17.9	17.9	17.9	17.9	17.8
Execution Time %	8.5%	17.0%	17.6%	19.3%	28.7%	29.2%	29.7%

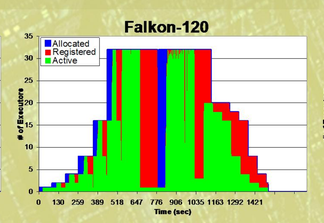
	GRAM +PBS	Falkon-15	Falkon-60	Falkon-120	Falkon-180	Falkon-∞	Ideal (32 nodes)
Time to complete (sec)	4904	1754	1680	1507	1484	1276	1260
Resouce Utilization	30%	89%	75%	65%	59%	44%	100%
Execution Efficiency	26%	72%	75%	84%	85%	99%	100%
Resource Allocations	1000	11	9	7	6	0	0

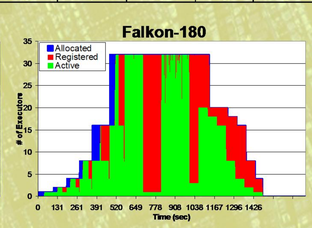










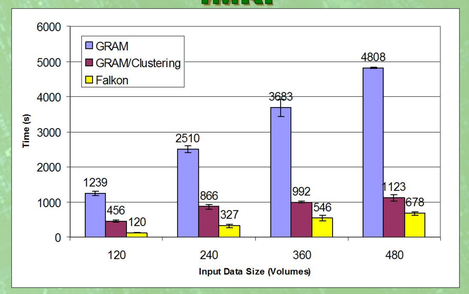


Applications

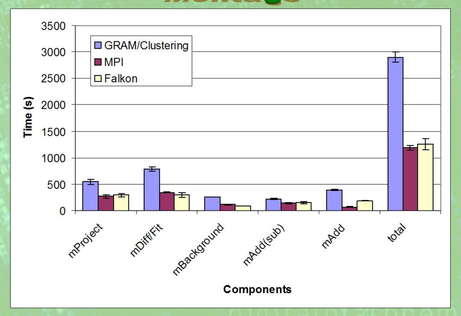
Swift Applications

Application	#Jobs/workflow	#Levels
ATLAS: High Energy Physics Event Simulation	500K	1
fMRI DBIC: AIRSN Image Processing	100s	12
FOAM: Ocean/Atmosphere Model	2000	3
GADU: Genomics	40K	4
HNL: fMRI Aphasia Study	500	4
NVO/NASA: Photorealistic Montage/Morphology	1000s	16
QuarkNet/I2U2: Physics Science Education	10s	3 ~ 6
RadCAD: Radiology Classifier Training	1000s	5
SIDGrid: EEG Wavelet Processing, Gaze Analysis	100s	20
SDSS: Coadd, Cluster Search	40K, 500K	2, 8

fMRI



Montage



Related Material, Further Reading, and Acknowledgements

Ioan Raicu, Yong Zhao, Catalin Dumitrescu, Ian Foster, Mike Wilde. "*Falkon: a Fast and Light-weight tasK executiON framework*", under review at IEEE/ACM SuperComputing 2007.
Ioan Raicu, Ian Foster, Alex Szalay, Gabriela Turcu. "*AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis*", TeraGrid Conference 2006, June 2006.
Ioan Raicu, Ian Foster, Alex Szalay. "*Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets*", poster presentation, IEEE/ACM SuperComputing 2006.
Yong Zhao, Mihael Hategan, Ben Clifford, Ian Foster, Gregor von Laszewski, Ioan Raicu, Tiberiu Stef-Praun, Mike Wilde. "*Swift: Fast, Reliable, Loosely Coupled Parallel Computation*", to appear at IEEE Workshop on Scientific Workflows 2007.

For more information, please visit http://people.cs.uchicago.edu/~iraicu/research/ and http://people.cs.uchicago.edu/~iraicu/research/AstroPortal/ and http://people.cs.uchicago.edu/~iraicu/research/Falkon

This work was funded from the NSF TeraGrid grant from June 2005 - September 2006; the work is currently being funded by the NASA ARC GSRP from October 2006 - September 2008.