Mixing Cloud and Grid Resources for Many Task Computing

David Abramson

Monash e-Science and Grid Engineering Lab (MeSSAGE Lab)
Faculty of Information Technology

Science Director: Monash e-Research Centre

ARC Professorial Fellow
Introduction

- A typical MTC Driving Application
- The Nimrod tool family
- Things the Grid ignored
  - Deployment
  - Deadlines (QoS)
- Clusters & Grids & Clouds
- Conclusions and future directions
A Typical MTC Driving Application
A little quantum chemistry
Wibke Sudholt, Univ Zurich

\[ U_{\text{eff}}(r) = A_1 \exp(-B_1 r^2) + A_2 \exp(-B_2 r^2) \]
SCO3 testbed
The Nimrod Tools Family
Nimrod supporting “real” science

- A full parameter sweep is the cross product of all the parameters (Nimrod/G)
- An optimization run minimizes some output metric and returns parameter combinations that do this (Nimrod/O)
- Design of experiments limits number of combinations (Nimrod/E)
- Workflows (Nimrod/K)
Plan File

Nimrod Portal

Nimrod/O

Nimrod/E

Nimrod/G

Actuators

Grid Middleware

### Parameter Section

- **Temp** label: float, range from 200 to 300, points 2
- **Pressure** label: float, range from 5000 to 8000, points 4
- **Concent** label: float, range from 0.002 to 0.005, points 2
- **Material** label: text, select any of "Fe", "Al"

### Tasks Section

- task rootstart
- task nodestart
- task main
- task nodefinish
- task roottfinish

Add a new parameter | Add a comment
---|---

Add a new task

Save | Cancel and Reload | Text mode
Nimrod Development Cycle

- Prepare Jobs using Portal
- Jobs Scheduled Executed Dynamically
- Results displayed & interpreted
- Sent to available machines
Nimrod Portal
Welcome Blair Bethwaite (you are an administrator)

Status of test

The experiment has executed for 8mins and 0secs.
The experiment has started, 29 jobs have been completed, there are 23 jobs waiting, 38 jobs executing and 0 jobs failed.

Pause Experiment

Plan file

```plaintext
parameter x integer range from 7 to .1 step 1;
parameter y float range from 0.125 to 1.25 step 0.125;

task main
  copy work.py node:
  node:execute /bin/chmod 4x work.py
  node:execute /bin/hostname >> output
  node:execute /bin/date >> output
  node:execute /bin/pwd >> output
  node:execute /bin/echo 'working...' >> output
  node:execute ./work.py $x() $y() >> output
  node:execute output output.f$(jobname).x$(x).y$(y)
endtask
```

You cannot change the plan after execution has started.
Multi-Task Computing
From drug to aircraft to antenna design
Nimrod/G Architecture

- Nimrod/G Client
- Nimrod/G Client
- Rsrc. Scheduler
- Nimrod/G GUI

Enfuzion API +

Database

- Level 3
- Level 2
- Level 1

Creator

Job Scheduler

Agent Scheduler

DB Server

Legion Actuator

Condor Actuator

Grid Middleware

Globus Actuator

Agent

RM & TS

Globus enabled node

Legion enabled node

RM: Local Resource Manager, TS: Trade Server

Condor enabled node.
Nimrod/K Workflows
Nimrod/K Workflows

- Nimrod/K integrates Kepler with
  - Massively parallel execution mechanism
  - Special purpose function of Nimrod/G/O/E
  - General purpose workflows from Kepler
  - Flexible IO model: Streams to files
Kepler Directors

- **Orchestrate Workflow**
- **Synchronous & Dynamic Data Flow**
  - Consumer actors not started until producer completes
- **Process Networks**
  - All actors execute concurrently
- **IO modes produce different performance results**
- **Existing directors don’t support multiple instances of actors.**
Workflow Threading

• Nimrod parameter combinations can be viewed as threads
• Multi-threaded workflows allow independent sequences in a workflow to run concurrently
  – This might be the whole workflow, or part of the workflow
• Tokens in different threads do not interact with each other in the workflow
The Nimrod/K director

• Implements the Tagged Data Architecture
• Provides threading
• Maintains copies (clones) of actors
• Maintains token tags
• Schedules actor’s events
MTC through Data Flow Execution
Dynamic Parallelism

Token Colouring

Diagram showing an actor connected to clone 1, clone 2, and clone 3.
So ...
Director controls parallelism

• Uses Nimrod to perform the execution
Complete Parameter Sweep

- Using a MATLAB actor provided by Kepler
- Local spawn
  - Multiple thread ran concurrently on a computer with 8 cores (2 x quads)
  - Workflow execution was just under 8 times faster
- Remote Spawn
  - 100’s – 1000’s of remote processes
Parameter Sweep Actor

Parameter Sweep

MatlabExpression-Nimrod

File Writer

Nimrod Director

Edit parameters for Parameter Sweep

- parameter K_MT_Nak type float range from 0.0001 to 0.008 points 2
- parameter K_MD_Nak type float range from 0.01 to 1 points 2
- parameter K_MgATP type float range from 1 to 2 points 2
- parameter K_MgATP_pca1 type float range from 0.0012 to 0.12 points 2
- parameter K_MgATP_pca2 type float range from 0.1 to 10 points 2
- parameter K_MgATP_pca2 type float range from 0.023 to 2.3 points 2
- parameter K_MgATP_SERCA type float range from 0.001 to 0.1 points 2
- parameter K_MgATP_SERCA1 type float range from 0.014 to 1.4 points 2
- parameter K_MgATP_SERCA2 type float range from 5.1 to 100 points 2

class:
org.monash.nimrod.ParameterSweep

semanticType00:
urn:lsid:localhost:onto:2:1#GridFunction
Partial Parameter Sweep
Nimrod/EK Actors

- Actors for generating and analyzing designs
- Leverage concurrent infrastructure
No actor parameters need setting

No difference from the parameter sweep actors
Parameter Optimization: Inverse Problems

- Domain Definer
- Points Generator
- Optimizer
- Constraint Enforcer
- Execute Model

\[ F(x, y, z, w, \ldots) \]
Nimrod/OK Workflows

- Nimrod/K supports parallel execution
- General template for search
  - Built from key components
- Can mix and match optimization algorithms
Things the Grid ignored
Resource Scheduling

• What’s so hard about scheduling parameter studies?
  - User has deadline
  - Grid resources unpredictable
    • Machine load may change at any time
    • Multiple machine queues
  - No central scheduler
• Soft real time problem
Computational Economy

- Without cost ANY shared system becomes un-manageable
- Resource selection on based pseudo money and market based forces
- A large number of sellers and buyers (resources may be dedicated/shared)
- Negotiation: tenders/bids and select those offers meet the requirement
- Trading and Advance Resource Reservation
- Schedule computations on those resources that meet all requirements
Nimrod’s Scheduler

Soft real-time scheduling problem
Execution @ AU Peak Time

![Graph showing jobs over time for different clusters and systems.]

- Linux cluster - Monash (20)
- Sun - ANL (5)
- SP2 - ANL (5)
- SGI - ANL (15)
- SGI - ISI (10)
Execution @ AU Offpeak

Time

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 3 4 7 8 10 13 15 17 19 21 23 26 28 31 32 35 37 39 41 43 46 48 50 53 55 57 60</td>
</tr>
</tbody>
</table>

- Linux cluster - Monash (5)
- Sun - ANL (10)
- SP2 - ANL (10)
- SGI - ANL (15)
- SGI - ISI (20)
Scheduling for Time Optimization

![Graph showing the number of tasks in execution over time for different systems.]

- Condor-Monash
- Linux-Prosecco-CNR
- Linux-Barbera-CNR
- Solaris/Ultas2-TTtech
- SGI-ISI
- Sun-ANL
Scheduling for Cost Optimization

![Graph showing the number of tasks in execution over time for different systems]
Deployment

- Has largely been ignored in Grid middleware
  - Globus supports file transport, execution, data access

- Challenges
  - Deployment interfaces lacking
  - Heterogeneity

Globus 4.0 Services

- Grid Deploy Aware Clients
- High Performance Virtualization

Your Java Service

- Deployment
- GRAM
- RFT
- Delegation
- Index
- Trigger
- Archiver
- CAS
- OGSA-DAI
- GTCP
Deployment Service

• Hide the complexity in installing software on a remote resource.
• Use local knowledge about
  - the instruction set,
  - machine structure,
  - file system,
  - I/O system, and
  - installed libraries
Towards a Grid Deployment Service
(Wojtek Goscinski)

1. Ant Build File
2. Application Files
3. Application Files
4. Un-configured Files
5. Managed Job Service (GRAM)
6. Instantiated Application
4. Configured Application

User Security Scope
Remote Host

Managed Job Service (GRAM)
DistAnt Service
Reliable File Transfer Service (GridFTP)
Globus User Hosting Environment
Our approach is runtime-internal.

Why do Java & .NET support web services, UI, security and other libraries as part of the standard environment?

Functionality is guaranteed.

Similarly, we aim to provide guaranteed HPC functionality.
Clusters & Grids & Clouds
Nimrod over Clusters

Jobs / Nimrod experiment

Nimrod

Actuator, e.g., SGE, PBS, LSF, Condor

Local Batch System
Nimrod over Grids

• **Advantages**
  - Wide area elastic computing
  - Portal based point-of-presence independent of location of computational resources
  - Grid level security
  - Computational economy proposed
    • New scheduling and data challenges
  - Virtualization proposed (Based on .NET!)

• **Leveraged Grid middleware**
  - Globus, Legion, ad-hoc standards
Leveraging Cloud Infrastructure

• Centralisation is easier
  - (Clusters vs Grid)

• Virtualisation improves interoperability and scalability
  - Build once, run everywhere

• Computational economy, for real
  - Deadline driven
    • “I need this finished by Monday morning!”
  - Budget driven
    • “Here’s my credit card, do this as quickly and cheaply as possible.”

• Cloud bursting
  - Scale-out to supplement locally and nationally available resources
Cloud Architectures

• IaaS
  - Build a virtual cluster

• PaaS
  - Leverage platform services

• SaaS
  - Nimrod portal installed on cloud
Integrating Nimrod with IaaS

Portal → Jobs / Nimrod experiment → Nimrod-O/E/K

Nimrod/G

Actuator: Globus,...  Services  New actuators: EC2, Azure, IBM, OCCI?,...?

Grid Middleware
Agents

RESTful IaaS API

VM Agents

VM Agents

VM Agents
Integrating Nimrod with IaaS

- Nimrod is already a meta-scheduler
  - Creates an ad-hoc grid dynamically overlaying the available resource pool
  - Don’t need all the Grid bells and whistles to stand-up a resource pool under Nimrod, just need to launch our code

- Requires explicit management of infrastructure

- Extra level of scheduling - when to initialise infrastructure?
Integrating Nimrod with IaaS
PaaS is trickier...

- More variety
  - Azure vs AppEngine
- Designed for web-app hosting
  - Nimrod provides a generic execution framework
- Higher level PaaS too prescriptive
  - AppEngine: Python and Java only
Nimrod-Azure Mk.1

- Nimrod server runs on a Linux box external to Azure
- Nimrod-Azure actuator module contains the code for managing Nimrod agents on Azure
  - pre-defined minimal NimrodWorkerService cspkg;
  - library for speaking XML over HTTP with the Azure Storage and Management REST APIs
Integrating Nimrod with Azure

To stand-up an Azure compute resource under Nimrod, the actuator:

- Copies the Nimrod agent package and encryption keys to an Azure Blob
- Adds command line parameters for agents to an Azure Queue
- Builds an initial cscfg for the deployment including relevant blob and queue URLs
- Deploys the service to the Cloud
Integrating Nimrod with Azure

Diagram showing the integration of Nimrod with Azure, including components like Nimrod Experiment, Azure Actuator, Nimrod Server, Blob, Queue, and cspkg.
Integrating Nimrod with Azure

Once deployed, the NimrodWorkerService:

- Pulls the Nimrod *agent* package from blobs referenced in cscfg settings
- Unpacks and launches the *agent* with parameters from the queue referenced by cscfg
- The *agent* connects out to the Nimrod server, pulling work and pushing results until: no work left; lifetime ends; exception
- But, when the *agent* exits there is no way to de-provision the role instance... scaling without de-scaling?! Please fix this!
Integrating Nimrod with Azure

Diagram showing:
- Blob
- Queue
- Azure Actuator
- Nimrod Server
- Agent
- User app/s
- Workers

Diagram arrows indicate connections between these components.
Grid + Amazon + Azure

Status of EC2vAzure

The experiment had executed for 1 hr, 54 mins and 44 secs.
All 8192 jobs have completed.

Check for grid errors

Archive or Delete the Experiment  Reset Experiment

Plan file

```plaintext
parameter time integer random from 20 to 60;
parameter jobs integer range from 1 to 8192 step 1;

task nodestart
    copy input model
    copy magic.bat.205 nodemagic.bat.xmel
endtask

task main
    nodestart
    substitute magic.bat.skel magic.bat
    output output
    execute chmod +x magic.bat
    error fail
    execute magic.bat > out
    execute hostname >> out
    copy output out/out.out.2jobsname
endtask
```
Conclusions and Future Directions

- **Commercial Clouds**
  - Grid economy == commercial clouds
  - Virtualisation built into fabric

- **Leverage MTC paradigm**
  - More complex Design of Experiments
  - More optimization Algorithms

- **Make environment more useful**
  - New portal
  - Workflows that interact with IO devices and Portals
Questions?

More information:

http://messagelab.monash.edu.au
Acknowledgements
MeSSsAGE Lab

- Faculty Members
  - Jeff Tan
  - Maria Indrawan
- Research Fellows
  - Blair Bethwaite
  - Slavisa Garic
  - Jin Chao
- Admin
  - Rob Gray
- Current PhD Students
  - Shahaan Ayyub
  - Philip Chan
  - Colin Enticott
  - ABM Russell
  - Steve Quinette
  - Ngoc Dinh (Minh)
- Completed PhD Students
  - Greg Watson
  - Rajkumar Buyya
  - Andrew Lewis
  - Nam Tran
  - Wojtek Goscinski
  - Aaron Searle
  - Tim Ho
  - Donny Kurniawan
  - Tirath Ramdas

- Funding & Support
  - Amazon
  - Axceleon
  - Australian Partnership for Advanced Computing (APAC)
  - Australian Research Council
  - Cray Inc
  - CRC for Enterprise Distributed Systems (DSTC)
  - GrangeNet (DCITA)
  - Hewlett Packard
  - IBM
  - Microsoft
  - Sun Microsystems
  - US Department of Energy