A Decoupled Scheduling Approach for the GrADS Program Development Environment

DCSL

Ahmed Amin

Outline

- Introduction
- Related Work
- Scheduling Architecture
- Scheduling Algorithm
- Testbench
- Results
- Conclusions
- Future work

Problem Definition

- Distributed parallel computation
- Grid Environment

  →Grid Application Development Software (GrADS) [3]
- Grids are heterogeneous systems that allow sharing of resources (cpu, mem, storage, n/w, ...)
- Grids are autonomous

Goal

- We require scheduling of application tasks onto available resources
- Minimize execution time of application
- ***Decouple*** scheduling from the application

  - Two main components:
    - *Performance model*: Analytical metric for predicting application execution times on a given set of resources
    - *Mapper*: Provides directives for mapping logical application data / tasks to physical resources

Related Work

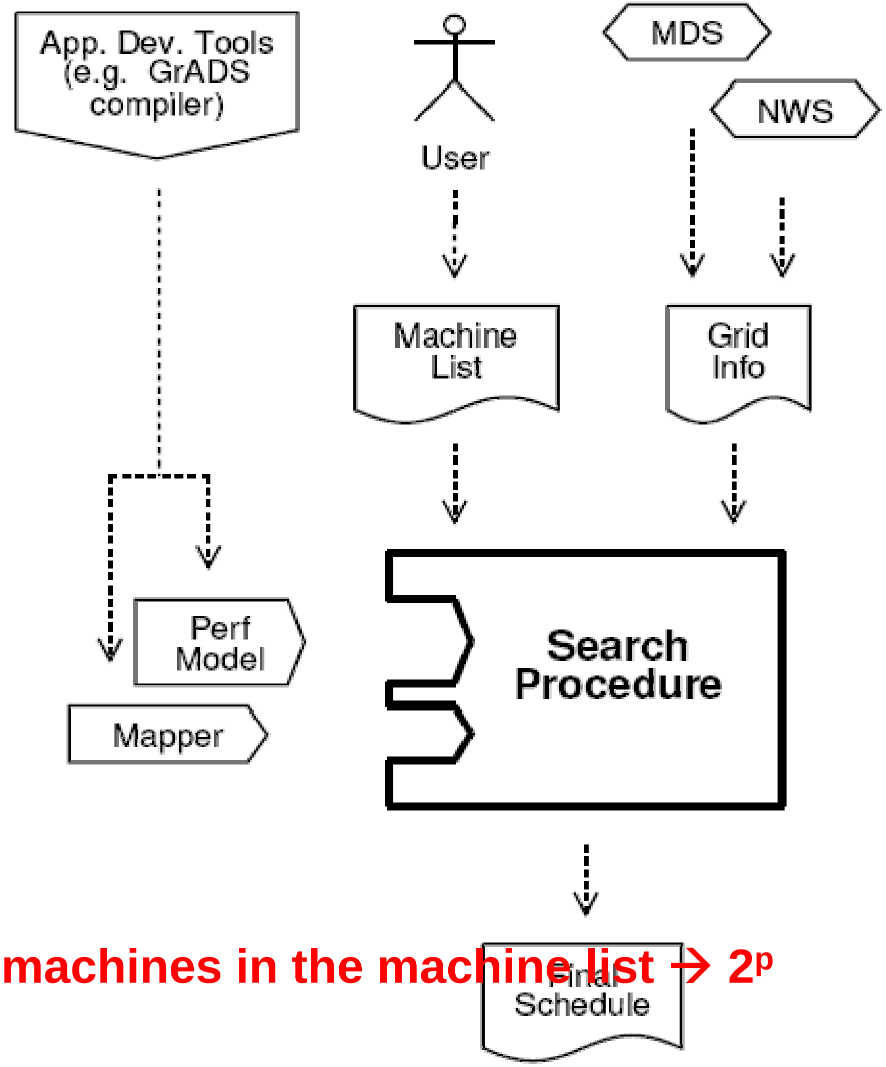
- AppLeS Project [4]:
  - Embeds scheduling logic into application
  - Or embed application specific information into the scheduler
  - Not easily retargeted
- Condor Matchmaker [5]
  - Single processor
- Prophet [6]
  - Dynamic run-time scheduling on heterogeneous systems
  - Must be written in Mentat programming language

Related Work

- Meta-scheduler designs:
  - Application specific schedulers are used to determine schedules for each application
  - Meta-scheduler evaluates performance of multiple application in the system
  - Meta-scheduler modifies the application specific schedules to improve performance

Scheduling Architecture

- MDS: Meta-computing Directory Service
  - (OS)
  - (CPU speed)
  - (# CPUs)
- NWS: Network Weather System
  - (% available CPU cycles)
  - (Free mem)
  - (BW)



An exhaustive search of p machines in the machine list → 2^p

Scheduling Algorithm

- *Schedule*: ordered list of machines and a mapping of data / tasks to those machines
- Desirable individual machine characteristics
- Desirable aggregate characteristics
  - E.g. fast machines with low network latency between them

Scheduling Algorithm

3p2^{s} CMGs [2]

```
Algorithm : SCHEDULESEARCH(machList, gridInfo, PerfModel, Mapper)

sites ← FindSites(machList)
siteCollections ← ComputeSiteCollections(sites)
for each collection in siteCollections
   for each machineMetric in (computation, memory, dual)
      for targetSize ← 1 to size(collection)
         list ← SortDescending(collection, machineMetric)
         CMG ← GetFirstN(list, targetSize)
         currSched ← GenerateSchedule(CMG, Mapper, PerfModel)
         if currSched.predTime < bestSched.predTime
           bestSched ← currSched
return (bestSched)
```

Scheduling Information

- Obtained from MDS, NWS

- Static
- Dynamic
- User

Testbench

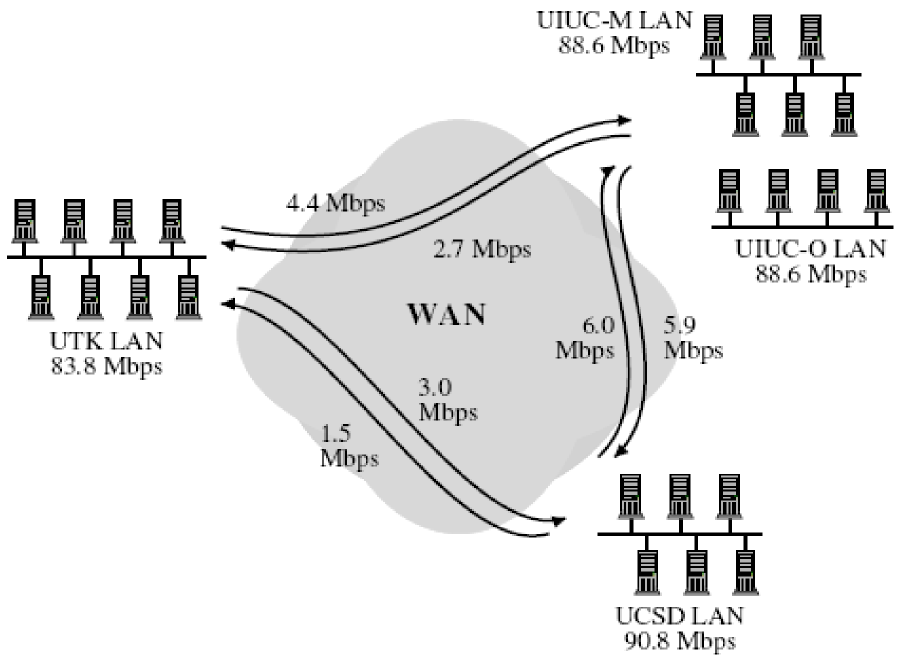
- Game of Life
- Jacobi
- Problem size N

} SPMD

- C & MPI / MPICH-G
- Performance Model: simple computation to predict iteration times
- Mapper:
  - Time-balance machine utilization (minimize idle times of selected machines)
  - Constraint optimization → lp_solve

Testbed

- One-site testbed
  - N = {600, 1200, 2400, 4800, 7200, 9600}
- Three-site testbed
  - N = {600, 4800, 9600, 14400, 16800, 19200}
- NWS “up” vs NWS “down”



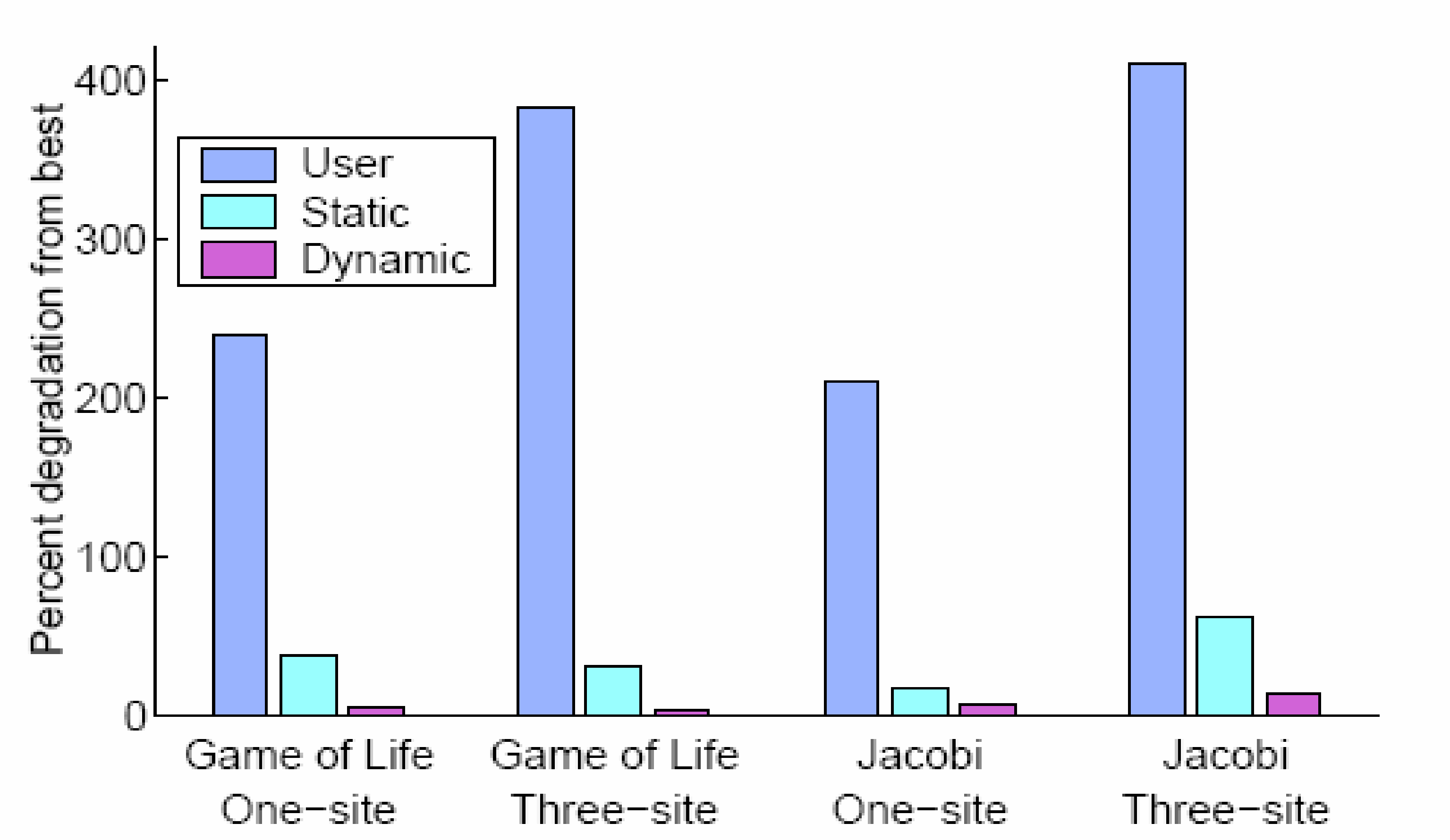
NWS capture

Experimental Scenarios

- Dynamic, Static, User
- 104 iterations per application
- 2 app * 2 testbeds * 6 input * 3 strategies

$$\deg FromBest = \frac{itTime - itTime_{best}}{itTime_{best}} \times 100$$

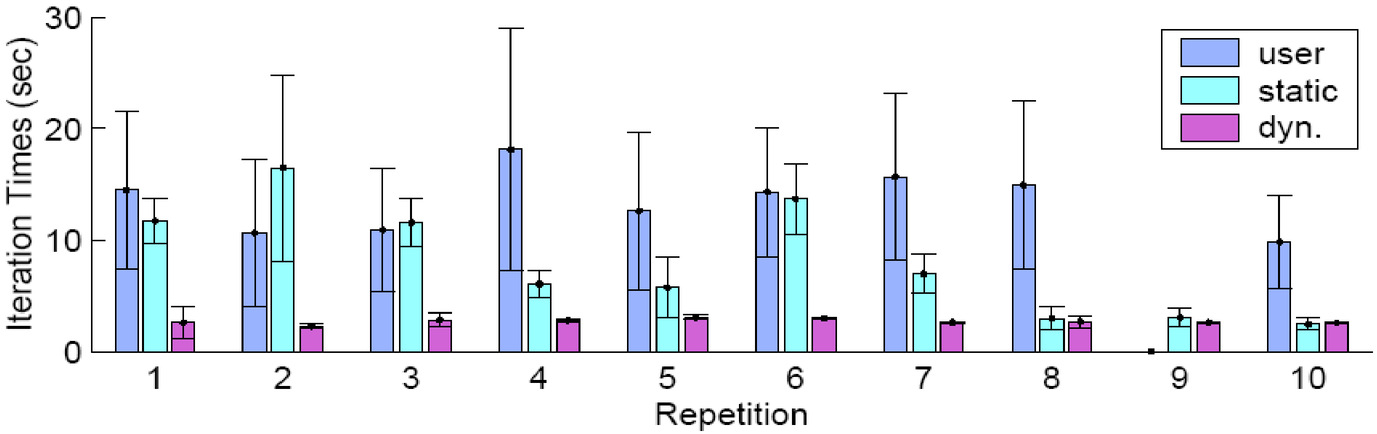
Results

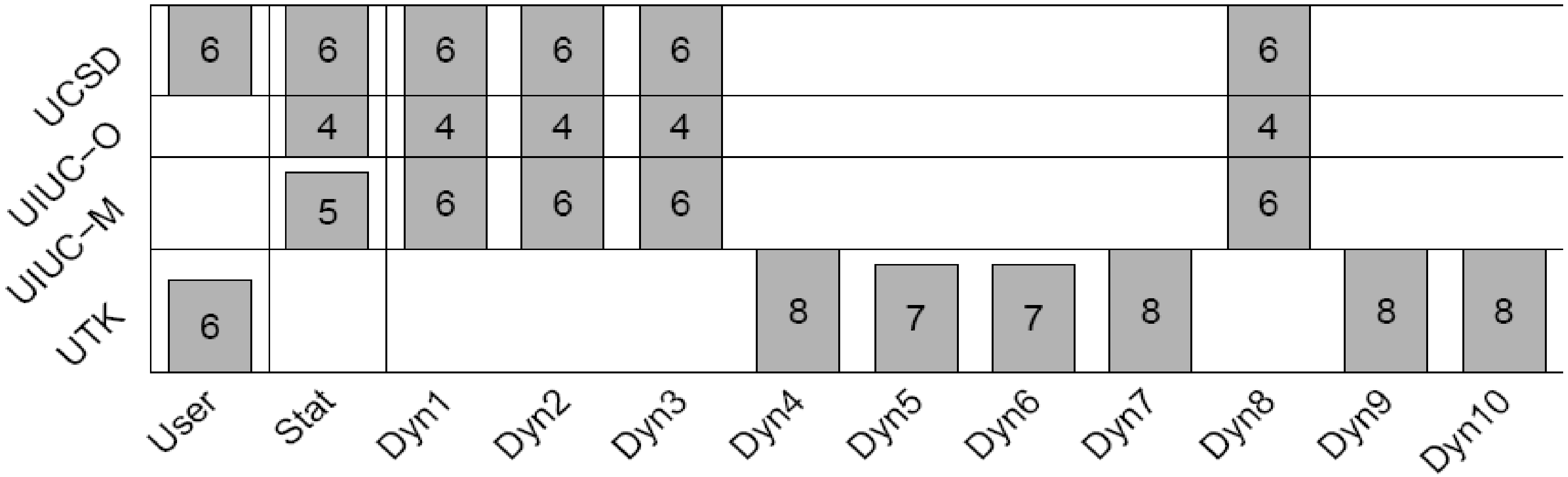

Percent degradation from best
400
300
200
100
0
User
Static
Dynamic
Game of Life
One−site
Game of Life
Three−site
Jacobi
One−site
Jacobi
Three−site

Results

	Game of Life - 1 site			Game of Life - 3 site			Jacobi - 1 site			Jacobi - 3 site		
	User	Stat	Dyn	User	Stat	Dyn	User	Stat	Dyn	User	Stat	Dyn
Avg	240.0	37.3	5.1	381.9	30.8	3.8	210.3	17.2	5.7	410.3	61.3	12.7
Std	152.0	40.4	12.9	466.6	63.3	10.7	130.6	28.2	12.6	212.7	145.8	40.6
Min	7.7	0	0	45.3	0	0	16.4	0	0	0	0	0
Max	507.7	156.9	69.3	2748.0	421.8	68.5	466.4	90.5	69.7	862.9	739.2	215.1

Case Study - Jacobi





Scheduling Overhead

Configuration	Info retr. (s)	Total (s)
Local MDS, NWS server	2.0	4.5
Remote NWS server	59.6	62.4
Remote MDS server	1087.5	1088.4

Conclusion

- Application has been decoupled from scheduling
- Reduced execution time compared to user defined schedules
- Exploits dynamic information at run time for improved scheduling

Future Work

- Extend performance evaluation on parameters other than execution time
- Compiler generated performance models
- Mapping strategies

References

- [1] H. Dail, H. Casanova and F. Berman "A Decoupled Scheduling Approach for the GrADS Program Development Environment" 2002
- [2] H. Dail " A modular framework for adaptive scheduling in grid application development environments" 2002
- [3] LIU, C., YANG, L., FOSTER, I., AND ANGULO, D. "Design and evaluation of a resource selection framework for Grid applications" 2002
- [4] BERMAN, F., WOLSKI, R., FIGUEIRA, S., SCHOPF, J., AND SHAO, G. "Application level scheduling on distributed heterogeneous networks" 1996
- [5] RAMAN, R., LIVNY, M., AND SOLOMON, M. "Matchmaking: Distributed resource management for high throughput computing" 1998
- [6] WEISSMAN, J. "Prophet: Automated scheduling of SPMD programs in workstation networks" 1999

	Circus machines	Torc machines	Opus machines	Major machines
Domain	ucsd.edu	cs.utk.edu	cs.uiuc.edu	cs.uiuc.edu
Nodes	6	8	4	6
Names	dralion, mystere soleil, quidam saltimbanco nouba	torc1, torc2 torc3, torc4 torc5, torc6 torc7, torc8	opus13-m opus14-m opus15-m opus16-m	amajor, bmajor cmajor, fmajor gmajor, hmajor
Processor	450 MHz PIII dralion, nouba 400 MHz PII others	550 MHz PIII	450 MHz PII	266 PII
CPUs/Node	1	2	1	1
Memory/Node	256 MB	512 MB	256 MB	128 MB
OS	Debian Linux	Red Hat Linux	Red Hat Linux	Red Hat Linux
Kernel	2.2.19	2.2.15 SMP	2.2.16	2.2.19
Network	100 Mbps shared ethernet	100 Mbps switched ethernet	100 Mbps switched ethernet	100 Mbps shared ethernet

- <> **What is Grid ?**
- In June'02, I attended the Grid Computing Planet conference in San Jose, California and I was surprised to learn that people even call clusters as grids. I believe that it is a marketing hype. Here is my definition of the Grid, which is based on my presentation as part of the "Understanding the Grid" panel:
- ***Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed "autonomous" resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements.***
- It should be noted that Grids aim at exploiting synergies that result from cooperation--ablity to share and agreegrate distributed computational capabilities and deliver them as service.
- **How is Grid different from other technologies such as Clusters / P2P / ASP? What about Grid Economy and Resource Management ? What about your work in this area ? and so on...**
- I have been asked such questions very frequently, which has trigged me to start creating this FAQ page.
- The key distinction between clusters and grids is mainly lie in the way resources are managed. In case of clusters, the resource allocation is performed by a centralised resource manager and all nodes cooperatively work together as a single unified resource. In case of Grids, each node has its own resource manager and don't aim for providing a single system view. Some of these points are being highlighted in my panel presentation at P2P 2002 conference. It should be noted that autonomous resources in the Grid can span across a single or multiple organisations.