Alternative Aspects of Cloud and Utility Computing

Vaidy Sunderam

*Emory University, Atlanta, USA*

vss@emory.edu
Credits and Acknowledgements

- **Distributed Computing Laboratory, Emory University**
  - Magda Slawinska, Jarek Slawinski, Vaidy Sunderam

- **Collaborators**
  - Oak Ridge Labs (A. Geist, C. Engelmann)
  - Univ. Tennessee (J. Dongarra, G. Bosilica)

- **Sponsors**
  - U. S. Department of Energy
  - National Science Foundation
  - Emory University
Overview

Metacomputing research
- Software systems for heterogeneous concurrent computing
  - PVM, IceT, H2O, Harness ...
- Focus on resource and communications management
  - Customize (condition) resource according to application needs
  - Support multiple (selectable) communication frameworks
  - Base programming model, but enable deploying others

Current project: Unibus
- Lightweight, self-organizing framework for metacomputing
- Relationship to Grid/Cloud/Utility computing
  - Resource sharing and aggregation to create a coherent concurrent computing environment
Unibus Goals

Key requirements/functionality

- Aggregation and sharing of diverse resources
  - Make compatible or unify to the extent possible/required
- Customizing resources as needed
  - Deploy most suitable (concurrent) computing environment

Additional self-organizing properties

- Adaptivity and resilience via dynamic resource management
- Reduced/localized global state
  - Provider-provider, provider-client

Central idea

- Client side overlay and virtualization for aggregation and unification
Resource Sharing Abstraction

- Providers own resources
- *Independently* make them available over the network
- Clients discover, locate, and utilize resources
- Resource sharing occurs between single provider and single client
  - Relationships may be tailored as appropriate
  - Including identity formats, resource allocation, compensation agreements
- Clients can themselves be providers
  - Cascading pairwise relationships may be formed
Unibus Approach

- **Coherence and Unification**
  - Device driver model: device = remote resource
  - Driver + handle (e.g. vnode) = access daemon + mediator
  - Interfaces suited to device (resource) class

- **Adaptivity and Dynamism**
  - Plug-and-play (event) model
  - Dynamic environment preconditioning
    - Similar to loading drivers / mounting filesystem

- **Reducing global state**
  - *Client*-side overlay, virtualizing resources via proxies
  - Similar devices grouped together at *interface* side
    - No coordination necessary at remote end
Unibus Model
A Basic Implementation: ZF-MPI

ZF-MPI Console

- Virtual machine assembly
- Data synchronization
- Compile and build (// execution of remote shell commands)
- Application launch

```
1 zf-mpi> add ft_mpi user1-sun@{lab6a,lab6b,lab6c,lab6d,compute}
2 zf-mpi> add ft_mpi user2-linux@{wembley,gobo,emily,sprocket}
3 zf-mpi> ft_mpi setNS compute //nameserver host
4 zf-mpi> ft_mpi add ALL
5 zf-mpi> ft_mpi console conf
6 zf-mpi> sync ~/NPB3.2.1/NPB3.2-MPI ~/zf-mpi/
7 zf-mpi> cd ~/zf-mpi/NPB3.2-MPI
8 zf-mpi> make bt NPROCS=8 CLASS=B
9 zf-mpi> mv bin/bt.B.8 $HARNESS_BIN_DIR/$HARNESS_ARCH/
10 zf-mpi> ft_mpi ftmpirun compute -np 8 -o bt.B.8 > log
11 zf-mpi> cat log | grep "Time in seconds"
12 zf-mpi> ft_mpi console haltall
```
Generalizing Automated Deployment

Client side

Provider side

EMORY UNIVERSITY
Summary

- Client-side virtualization and aggregation
  - Complete freedom and flexibility for providers
  - Dynamic conditioning & unification -> effective shared access to remote heterogeneous multidomain resources

(a) Virtual Organization Model
(b) Public Infrastructure Model
(c) Proposed Client-Side Overlay Model