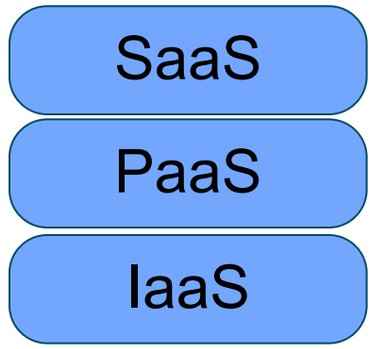
Eric Van Hensbergen & Phillip Stanley-Marbell - IBM Research
Noah Paul Evans - Alcatel-Lucent Bell-Labs
10 October 2009 - ACM SIGOPS LADIS 2009 Workshop

A Unified Execution Model for Cloud Computing

Motivation



Clouds have renewed interest in flexible distributed models of interacting with computer resources.

The true power of the cloud will only be realized by enabling cloud aware applications and cloud aware operating systems which are able to directly leverage the power and flexibility of the cloud as well as enabling reliability and scalability.

We believe that this is a systems problem best solved through standard interfaces and system calls provided by the operating system in order to not couple the facility to particular programming language or runtime middleware.

GOAL: break down the barriers between infrastructure management and traditional operating system resource management, creating a cohesive interface to request, manage and release resources from within the cloud.

Related Work

- Distributed Operating Systems
  - V, Amoeba, Eden, Cambridge Distributed Computing System, etc.
- XCPU & other HPC Distributed Middleware
- Snowflock
- PaaS
  - Google App Engine, Microsoft Azure

Implementation Overview

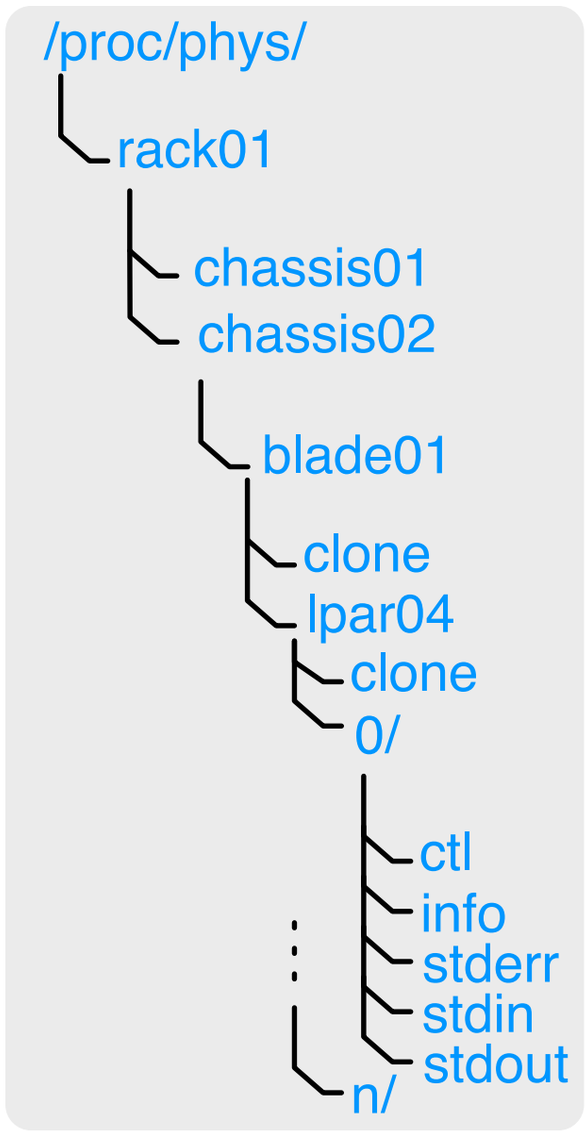
- Interfaces to provisioning, execution, monitoring and control maintained as a synthetic file system
  - –UNIX proc -> Plan 9 Proc -> XCPU Proc -> UEM
- Brasil Co-Operating System
  - –runs hosted on Linux, OSX, BSD, Windows, Plan 9
  - –exports interfaces into the native file system
  - –applications and runtimes can build library interfaces
- Can use Zeroconf to locate other nodes or can be parameterized to contact a “parent”
- Peer node UEM namespaces are union mounted allowing for transitive access to nodes on separate network segments via gateway nodes
- Modular Infrastructure with multiple levels of plug-ins
  - –Hierarchical Organization
  - –Policy Engines governing requests and manipulating in-flight resources
  - –Back-end resource providers (physical infrastructure, hypervisors, virtual machines)

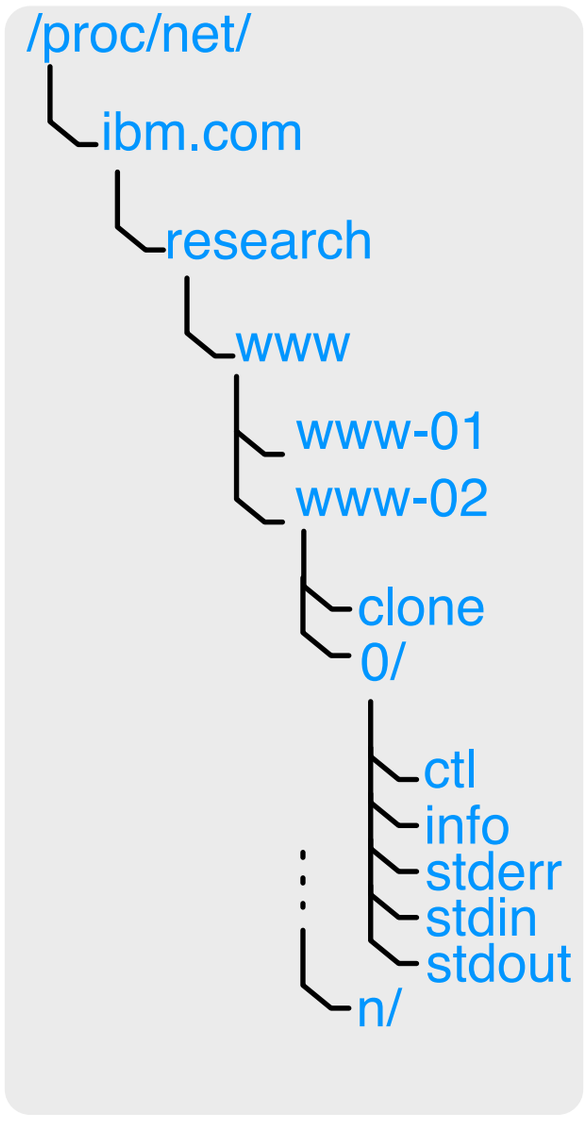
Synthetic File Systems

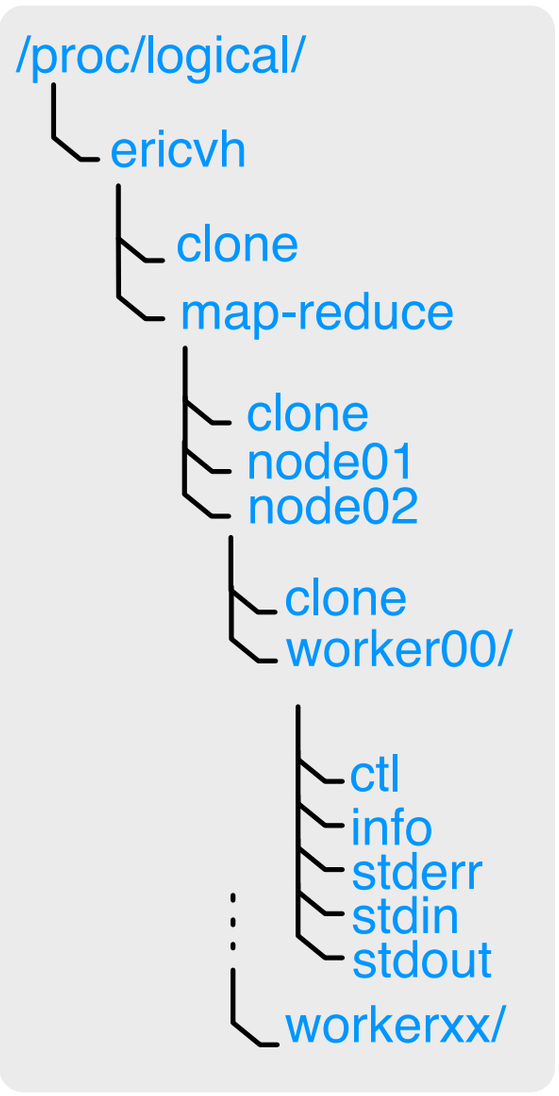
- All interaction is through a hierarchy of synthetic file systems
  - –language independent & easily distributable
  - –natural aggregation and security models
- Multiple hierarchical organizations with different scope rules are possible
  - –Canonical organization is typically based off network hierarchy
  - –Exploring task-based logical hierarchies combining logical nodes & threads
- Parent levels provided aggregate interfaces for controlling children
- Beyond static hierarchies - web inspired query paths can be used to perform provisioning and aggregate control

```
% ls /proc/query/x86/gpus=1/mem=4G
# will return a list of physical systems matching
0/ 1/ 2/
% echo kill > /proc/query/user=ericvh/ctl
# will terminal all LPARs and/or threads belonging
# to user ericvh
% echo cat /proc/query/os=linux/status
# returns status of all Linux logical partitions
node01 up 10:35, 4 users, load average: 1.08
node02 up 05:23, 1 users, load average: 0.50
node03 down ...
```

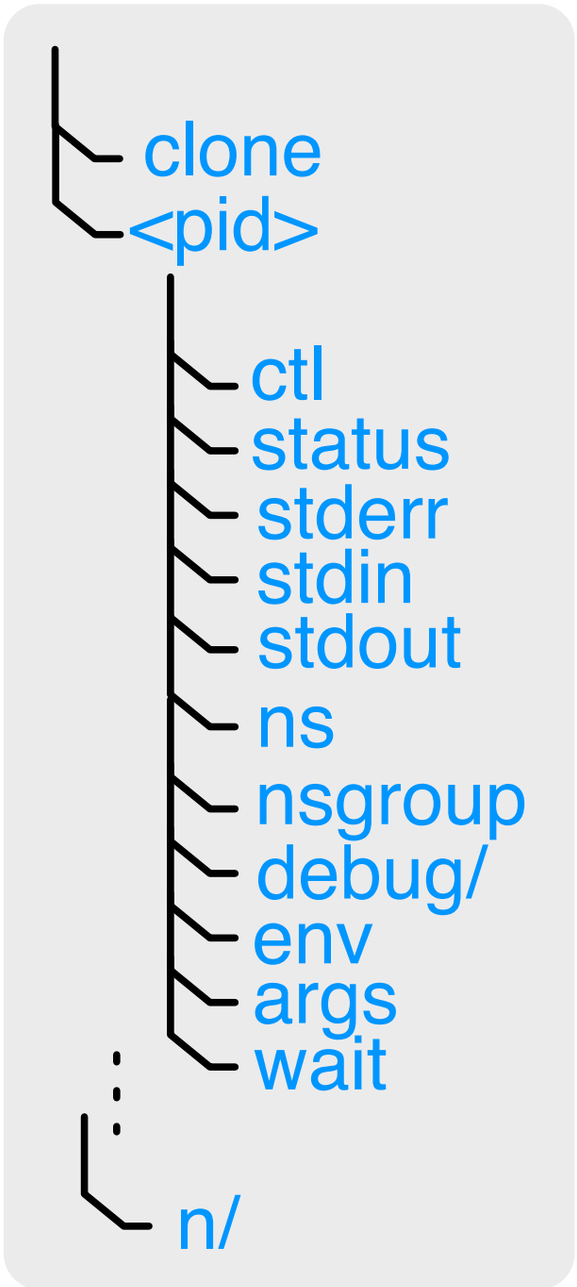
Different Organizational Models





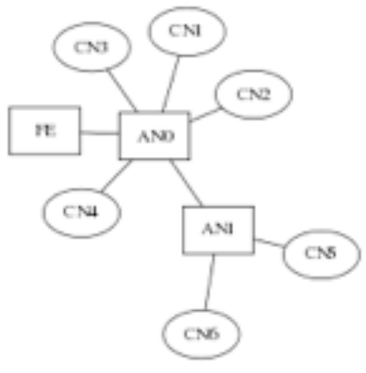


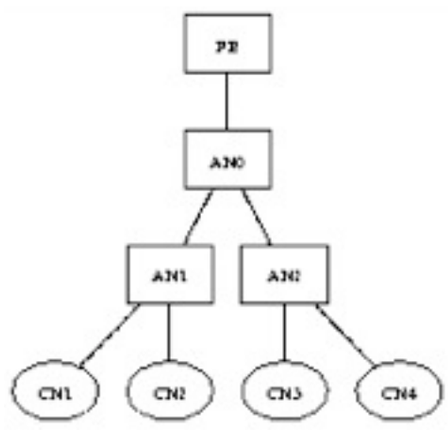
Execution & Control Mechanisms

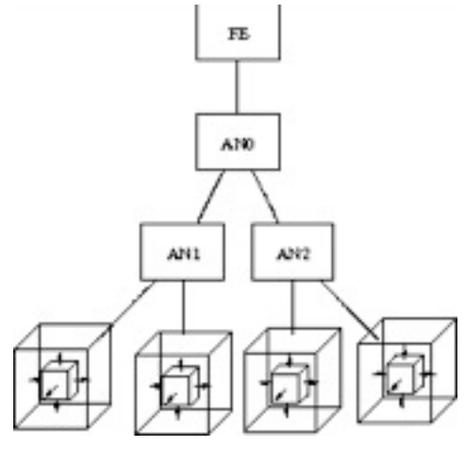


- At leaf-level, the interface is very similar to the XCPU2 [CLUSTERS 09] interface
- clone is used to establish a new instance
- ctl is a command console for controlling execution and other provisioning operations (triggering migration, network splicing, etc.)
- ns provides fs/storage namespace definitions
- env & args provide environment variables and execution arguments
- wait can be blocked upon to wait on program/ LPAR termination
- debug subdirectory provides direct access to memory, and as appropriate stack, registers, and could be leveraged to provide direct access to external debuggers (acid, gdb, etc.)
- stdin/stdout/stderr provide console for LPARS

Aggregation







- Large Scale Distributed Systems
  - –Scale is principle research challenge

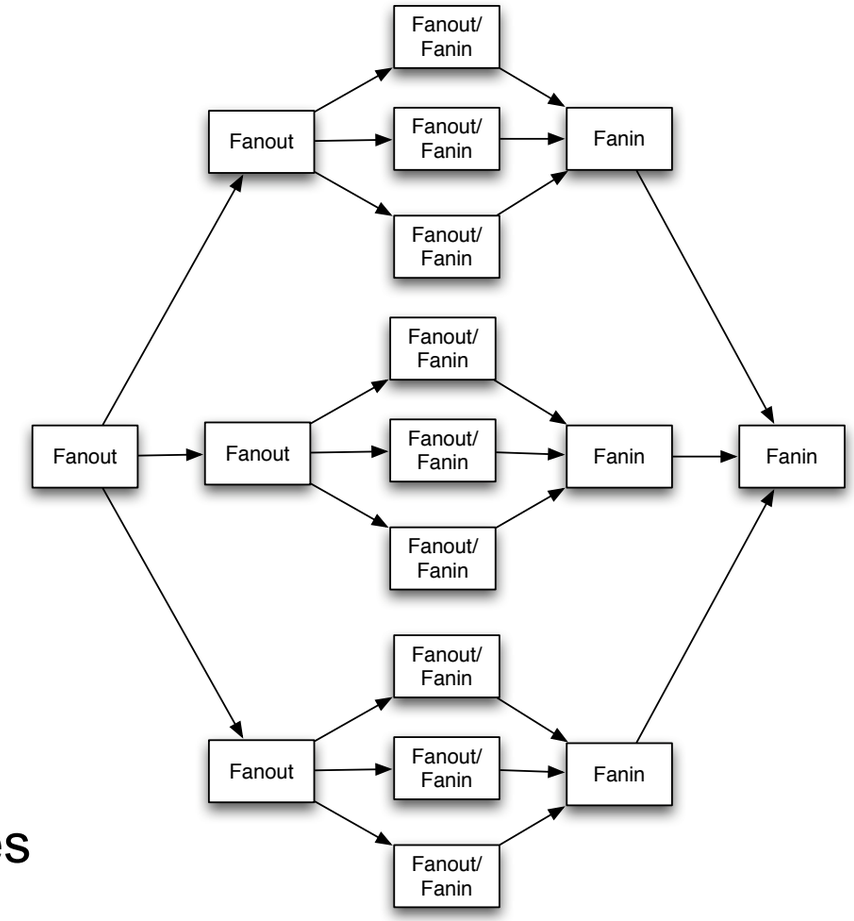
- Infrastructure topology best organized along natural aggregation points
  - –BG/P we have an I/O node for every 64 Compute Nodes
  - –Traditional Clusters can be organized by chassis or rack at similar density
  - –Modern virtualized environments typically have a Dom0 or Host as an aggregation point
  - –Multi-level dynamic aggregation also appealing for execution & other services [MTAGS08]

- Aggregated Synthetic File System Interfaces
  - –Information and Control Files at all levels of the hierarchy
  - –Information based synthetic files provided aggregated output
    - • can be XML/s-rec style aggregation, or could be application/task specific
  - –Commands sent to control files multiplexed to lower control files
    - • can leverage broadcast/multicast or other interconnect optimizations
  - –Allows altering granularity based on workload requirement

Abstracted Communications & Network Splicing

"This is the Unix philosophy. Write programs that do one thing and do it well. Write programs to work together." - Doug McIlroy

- UNIX Pipelines
  - –cat file | sort -n -r | uniq | more
- PUSH Multipipes [PODC09]
  - –ls |< cat | sort -n -r |#| uniq >| sort -n -r | more
  - –New shell operators
    - |< - Fan Out Pipe (One to Many)
    - |#| - Hash Pipe (Many to Many)
    - >| - Fan In Pipe (Many to One)



- Requires the ability to splice standard I/O between remote nodes
  - –need control abstractions within execution model to facilitate
- Example:

```
# proc102 | proc56 | proc256
# usage: splice <proc_path> <src_fd> <local_fd>
% echo splice /proc/net/remote.com/102 1 0 > /proc/net/mybox.com/56/ctl
% echo splice /proc/net/mybox.com/56 1 0 > /proc/net/farther.com/256/ctl
```

Policy Modules

- Pluggable
  - –Can be user-defined, application-defined, and administrator defined
- Protective
  - –keep rogue threads from allocating obnoxious number of resources
- Task driven resource allocation
  - –eliminate double clutch (allocate node, execute task)
  - –probably should avoid task migration (historically hasn't worked out)
    - • may be easier to migrate LPARs versus migrate tasks
  - –deriving resource requirements of a particular task may be difficult
- Resource Management
  - –Consolidate LPARs from underutilized hardware
  - –Colocate LPARs with heavy communication
- Resiliency
  - –spread out redundant LPARs to different availability domains

Status & Future Work

- Still undergoing rapid development/re-development
  - –Support for BlueGene/P both via Kittyhawk Linux & HARE Plan 9 Environments
  - –Preliminary support for EC2 compliant cloud APIs
- Will be released as open-source once some degree of stability is achieved
  - –http://www.research.ibm.com/hare
- Future Work
  - –Larger scale application deployments
  - –Integration with Eucalyptus environment
  - –Integration with a Virtual Machine as an additional level of the execution hierarchy
  - –Build a runtime environment on top of UEM
    - • Potentially combine UEM facilities with OpenCL or other language
  - –Integration of policy mechanisms/agents for load balancing, power aware, and migration based optimization
  - –Integration of abstracted support nodes (file servers, database servers, etc.)
  - –Need policy models for dynamically adapting in the face of failure, resource changes
  - –Heterogeneous Architectures - GPU & Cell Hybrid Systems [PPAC09]

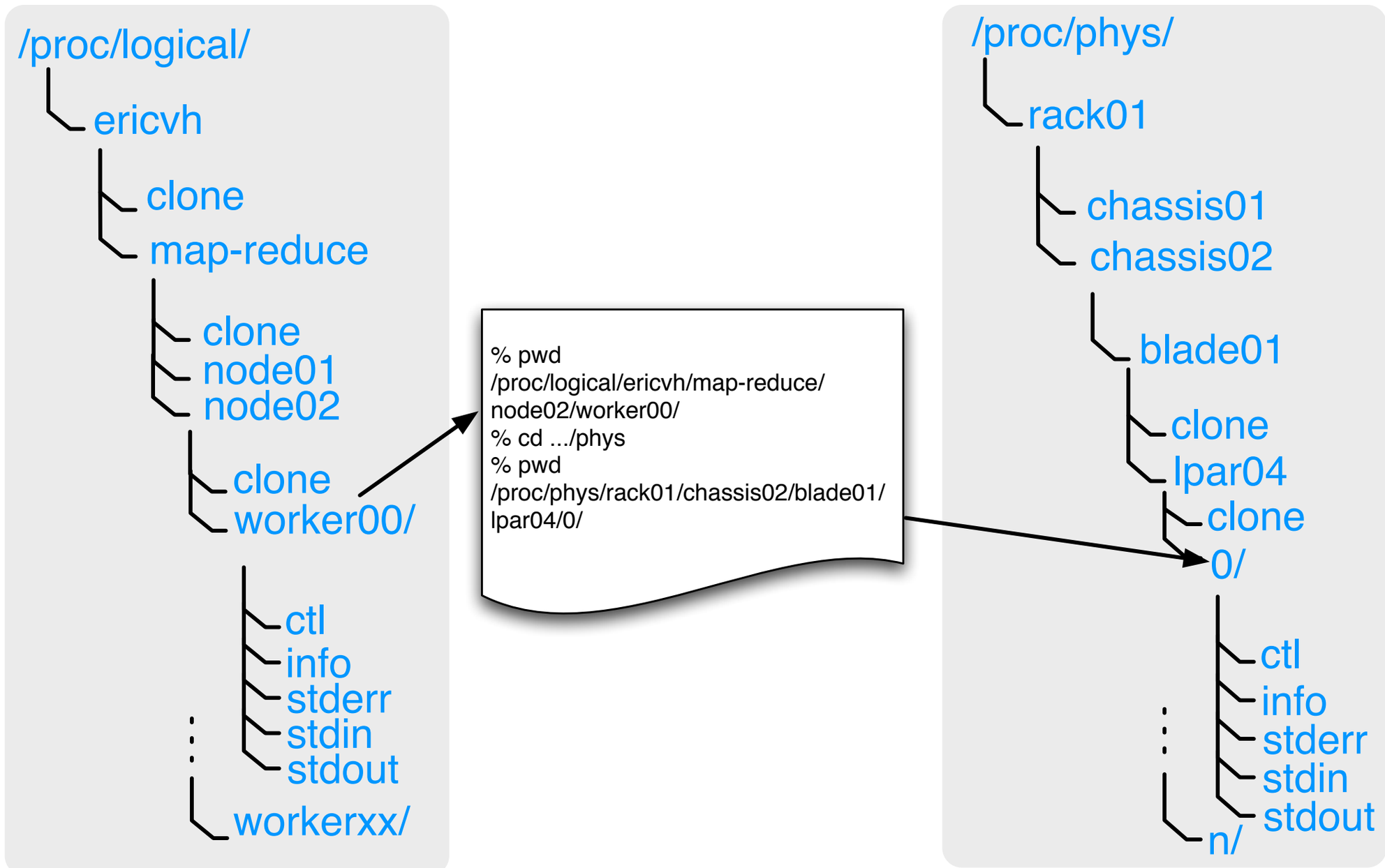
Acknowledgements

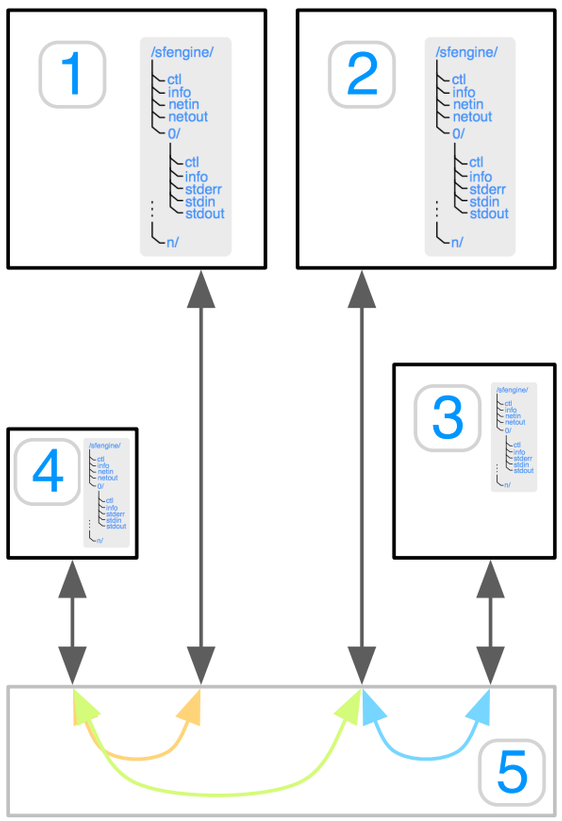
- This work has been supported in part by the Department of Energy Office of Science Operating and Runtime Systems for Extreme Scale Scientific Computation project under contract #DE-FG02-08ER25851

Thanks! Questions?

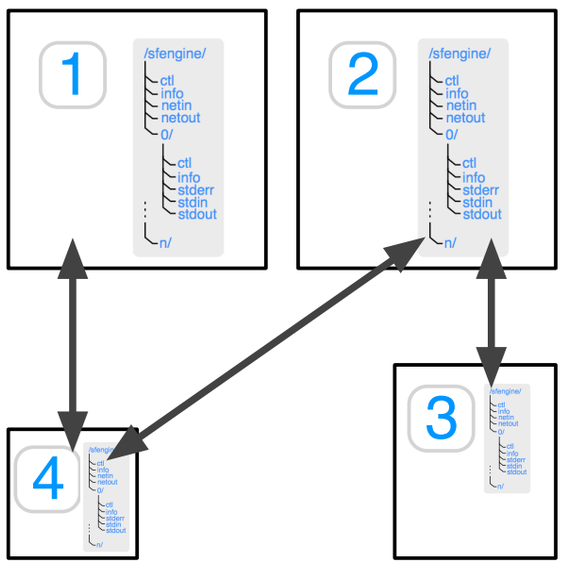
BACKUP SLIDES

dot-dot-dot namespace shortcuts





Central Splicing



Issues commands to UEMs of hosts 1– 4 to initiate splices. 5

Distributed Splicing

Time Evolving File System