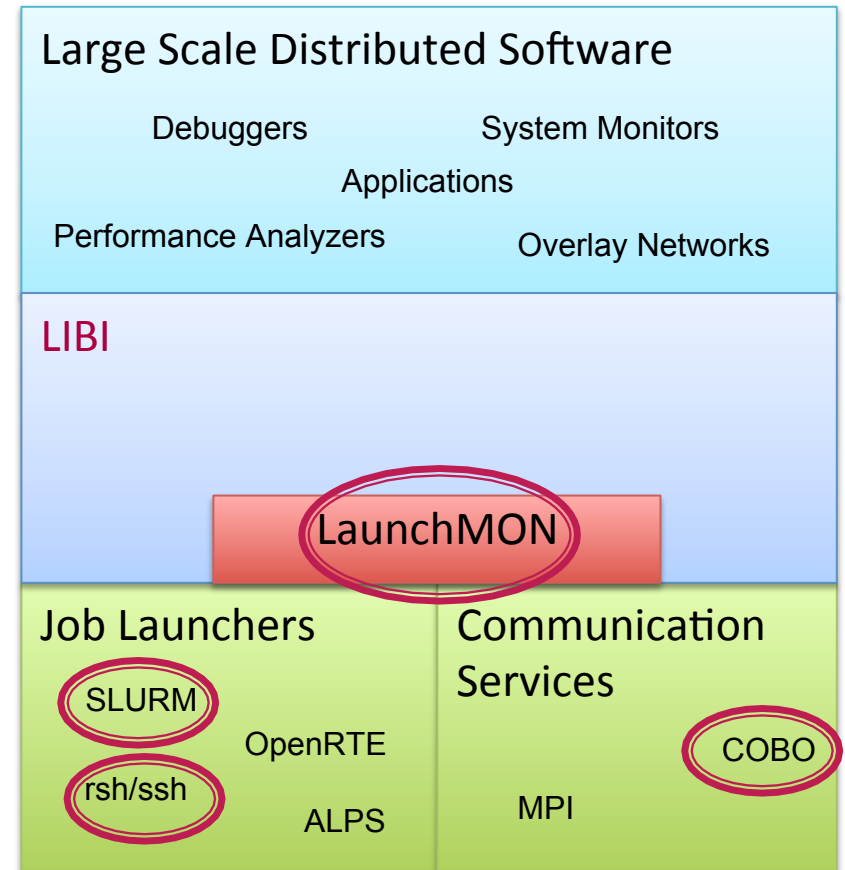


Toward exascale tool infrastructure (or what we've been up to this past year)

Dorian Arnold / *University of New Mexico*
With Taylor Groves and Whit Schonbein/ *University of New Mexico*

Where we were last year

- ▶ LIBI for normal MRNet startup
 - Optimal bulk process launch
 - Efficient propagation of initialization information
- ▶ What we desired
 - Handling MRNet's "disconnected startup" modes
 - Reducing topology specification burden



Today's adventures

Updates of our tool startup work:

1. Status of the MRNet/LIBI integration
2. Improving startup using scalable information services
3. An API for reduced tool topology specification



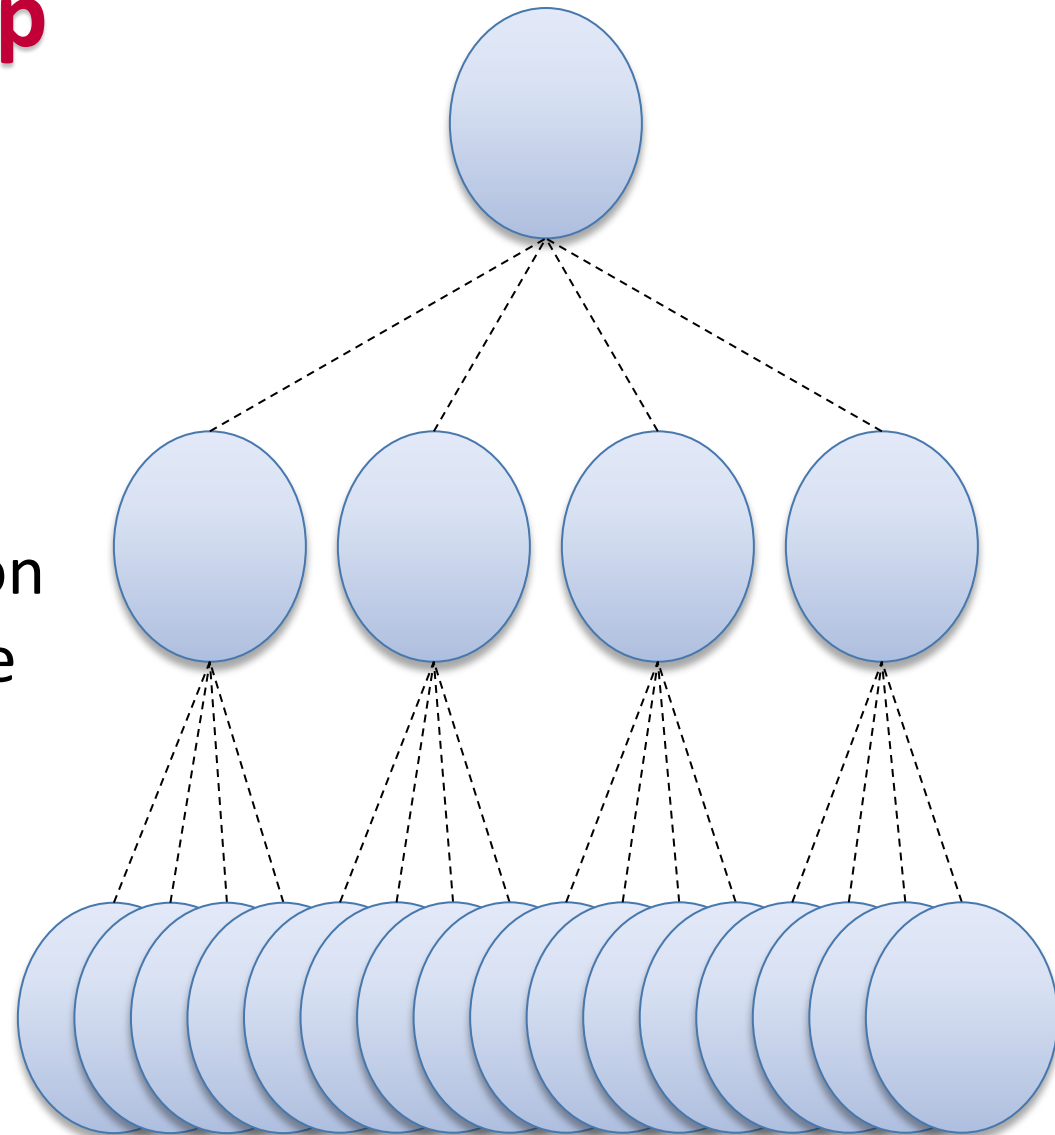
MRNet/LIBI Integration Status

- ▶ New LIBINetwork class
 - Previous network classes: RshNetwork and XTNetwork
 - Network class specifies MRNet's launching protocol
- ▶ Can use SLURM or rsh for process launch
 - `./configure --with-startup [libi-slurm|libi-ssh]`
 - Can still use old non-LIBI startup modes (but why? 😊)
- ▶ Tested against MRNet 4.0 (no regressions)
- ▶ Merged into MRNet master branch as of April 2014

Motivating scalable info. diss.:

Tree-based startup

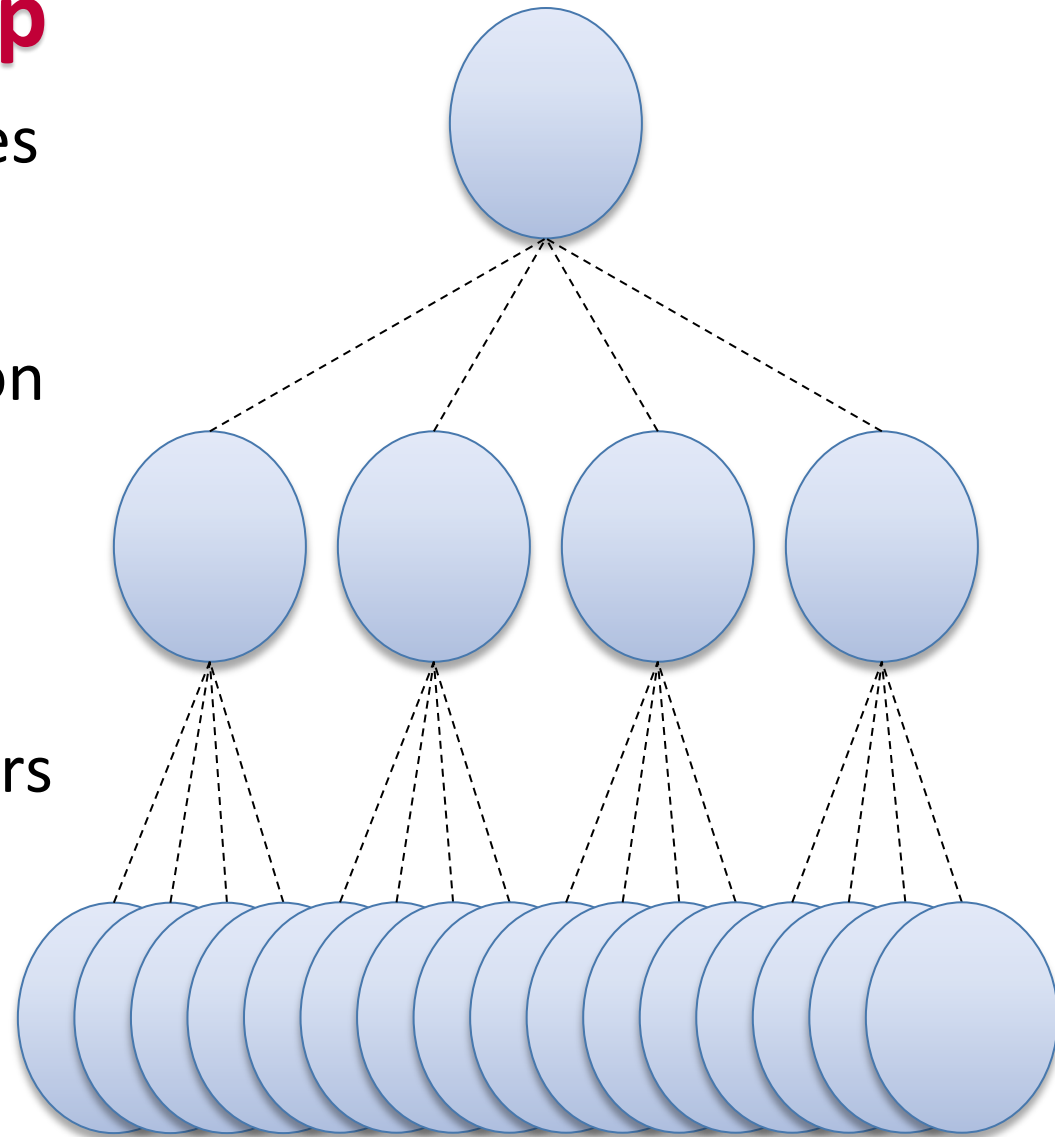
- ▶ Parent creates children
 - E.g. MRNet default
 - Local: `fork()/exec()`
 - Remote: `rsh/ssh`
- ▶ Configuration information passed via command line
- ▶ Requires starting all processes



Motivating scalable info. diss.:

Tree-based startup

- ▶ Root creates all processes
 - E.g. MRNet-LIBI
- ▶ Configuration information passed via custom mechanism
 - PMGR collectives
- ▶ Root gathers then scatters
- ▶ Requires starting all processes



What about disconnected startup?

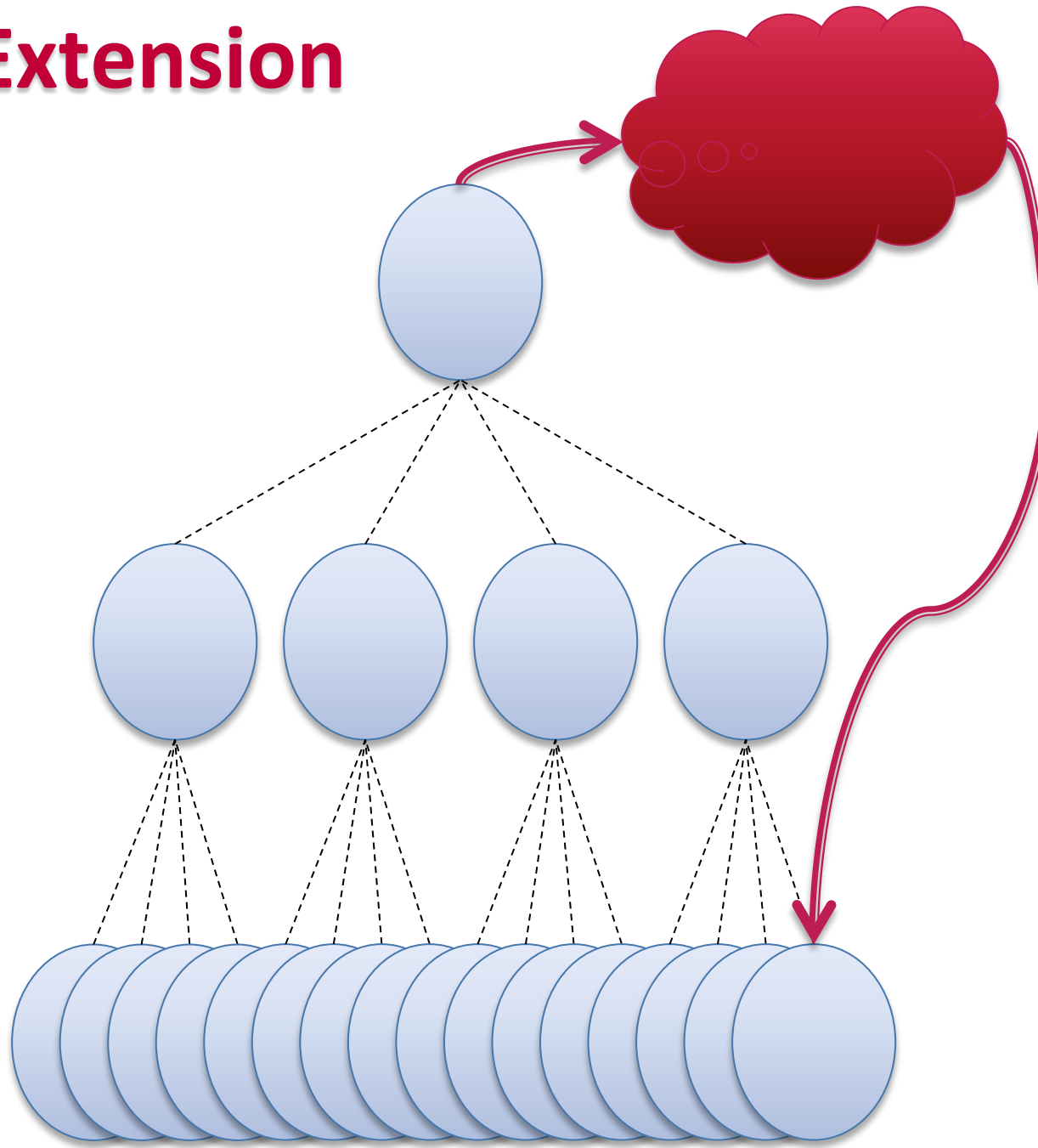
- ▶ Tool infrastructure does not start all processes
 - E.g. MRNet's "no back-end instantiation" & "lightweight back-end" modes
- ▶ How do back-ends learn and connect to parents?
 - Current solution: use the filesystem ☹️
- ▶ Why not leverage scalable information services (IS)?

Key-value stores to the rescue

- ▶ General MRNet extension for start-up data distribution
- ▶ Initial implementation uses MongoDB
 - A false start tried ZFS
- ▶ Prototype available in KVS branch of MRNet repository

MRNet KVS Extension

- ▶ Root creates internal nodes
 - ▶ Gathers configuration information
 - ▶ Publishes in KVS
- ▶ Third party creates leaves
- ▶ Leaves retrieve configuration information from KVS
- ▶ Leaves connect to internal nodes



New Node Discovery Engine (NDE)

- ▶ Generalized mechanism for
 - Initializing processes with target IS
 - Parents publishing startup information into IS
 - Orphans retrieving startup information from IS
- ▶ Parent and orphan interfaces

NDE: Node Information Object

- ▶ Hostname
- ▶ Port
- ▶ Rank
- ▶ Parent hostname
- ▶ Parent port
- ▶ Parent rank
- ▶ Session id //currently unused

MongoDB-based Prototype

Mongo-DB

- Open-source NoSQL database
- Written in C++

NDE: Example Front-end

...

```
//instantiate MRNet internal nodes as per usual
```

...

For all leaf internal nodes:

```
nodeinfo.iRank = (int)leaves[curr_leaf]->get_Rank();
```

```
nodeinfo.iport = (int)leaves[curr_leaf]->get_Port();
```

```
nodeinfo.ihostname = leaves[curr_leaf]->get_HostName();
```

```
//MongoParent is derived from NDEParent
```

```
MongoParent* parent = new MongoParent(info);
```

```
parent->set_DBHost(db);
```

```
parent->connect_toDB();
```

```
parent->send_MyNodeInfo();
```

NDE: Example Back-end

...

```
//MongoOrphan is derived from NDEOrphan
```

```
MongoOrphan orphan;
```

```
set_DBHost(&orphan, argv[1]);
```

```
connect_toDB(&orphan);
```

```
init_NDEO(&orphan, NULL, NULL);
```

```
discover_Parent(&orphan);
```

```
sprintf(parHostname, "%s", orphan.base.myInfo.phostname);
```

```
sprintf(parPort, "%d", orphan.base.myInfo.pport);
```

```
sprintf(parRank, "%d", orphan.base.myInfo.pRank);
```

```
sprintf(myHostname, "%s", orphan.base.myInfo.ihostname);
```

```
sprintf(myRank, "%d", orphan.base.myInfo.iRank);
```

```
//instantiate MRNet back-end node as per usual
```

...

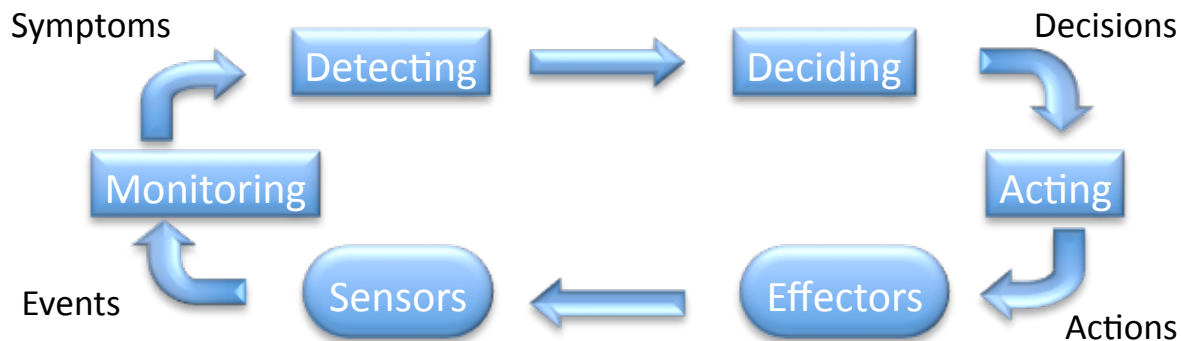
NDE to do list

- ▶ Instead of “root gather”, parents publish own data
- ▶ Comprehensive functionality testing
- ▶ Test performance to determine scalability
- ▶ Automatic peer/session discovery: investigate ways to avoid a priori known information, e.g. DB session IDs.
 - Persistent KVS services will help


A vision for autonomous TB̄ON infrastructure

TB̄ON Autonomy aka the self-* properties:

- ▶ **Self-configuring**
 - Automatic TB̄ON topology configuration
- ▶ **Self-monitoring**
 - TB̄ON health and performance
- ▶ **Self-healing**
 - TB̄ON Fault tolerance and failure recovery
- ▶ **Self-optimizing**
 - Dynamic TB̄ON reconfiguration to improve performance




Overall autonomous operation

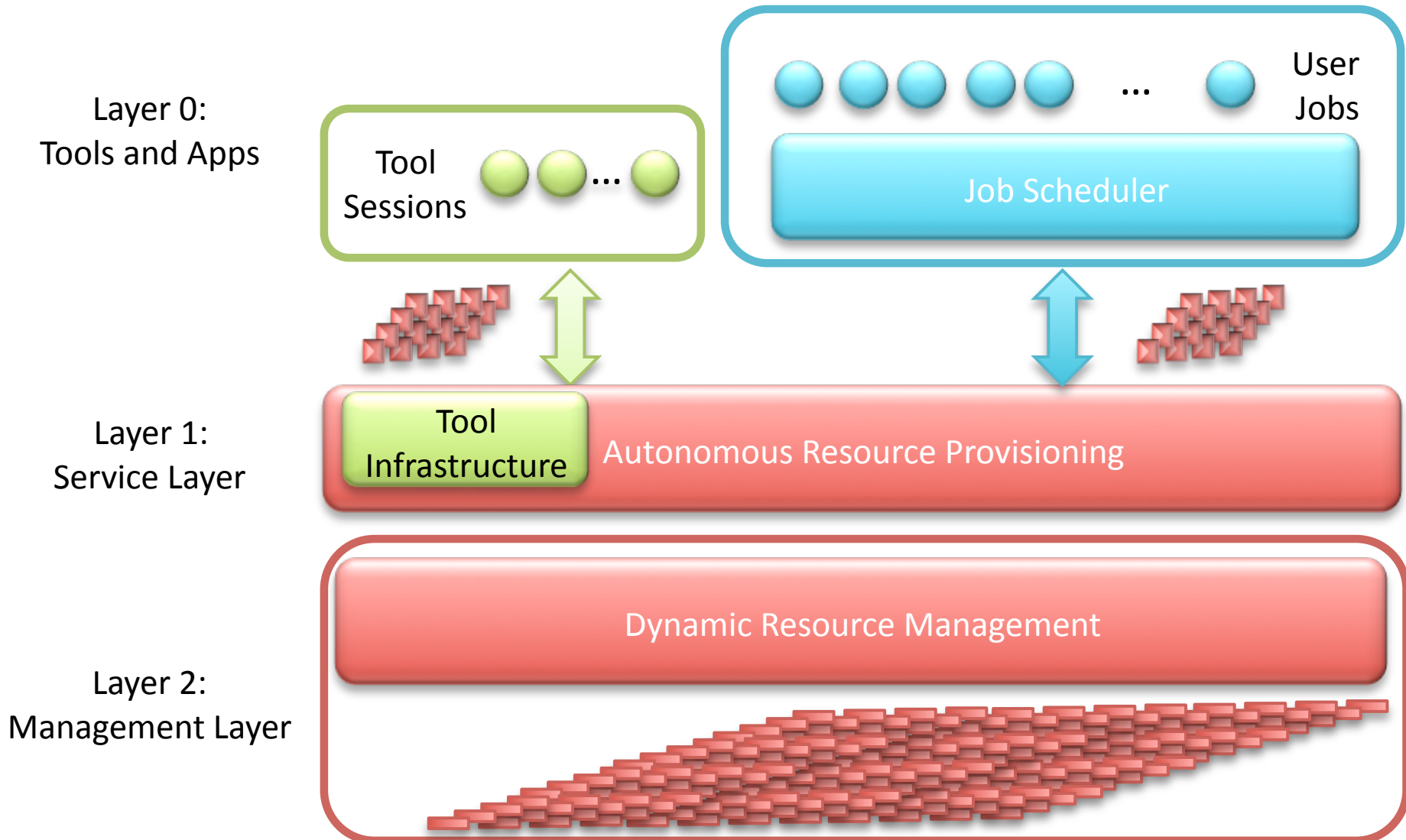
1. Collect metrics relevant to overlay performance
 2. Performance models diagnose performance failures
 3. Performance failure?
Heuristics for topology reconfiguration
 4. Find reconfiguration cost (overhead)/benefit (speedup)
 5. Reconfigure overlay when benefits outweigh costs
 6. Go to step 1
- 

Autonomous operation: Detecting performance failures

Generally, a sub-optimal overlay network topology

- Resource oversubscription: insufficient resources for offered workload
 - Resource undersubscription: insufficient work for allocated resources
 - Suboptimal configuration: resources not being effectively utilized
-
- ▶ Develop performance models
 - Coarse-grained approach
 - Consider processors and networks influence on topology performance
 - Must be accurate yet tractable to execute (potentially multiple times)
 - ▶ Build sensors to collect data to parameterize models
 - ▶ Compare current observed performance to other configurations
- 

A new dynamic ecosystem



Reducing Topology Specification: A step to auto-topology management

- ▶ Currently MRNet user specifies complete topology
 - Mapping of all processes to nodes
 - Interconnectivity amongst all processes
- ▶ Ideal: User specifies **nothing** about the topology
 - At least nothing about the internal tree topology
 - Front-end and back-end may be fixed based on usage
- ▶ Intermediate point: user gives generic topology information; we auto-configure the specifics

Auto-topology specification

- ▶ Physical (internal) nodes
 - User specifies
 - Eventual goal: MRNet requests from resource manager
- ▶ Process placement & inter-process connectivity
 - Static
 - Coarse policy and heuristic-based mappings
 - Eventual goals: dynamic, performance optimizations based on workload and physical network

Auto-topology specification methods

- ▶ `createTopologySpecification()`:
 - Inputs:
 - Front-end host
 - List of internal node hosts
 - List of back-end hosts
 - Topology specification policy:
 - Dictates the constraints to use for tree topology
 - Outputs: MRNet topology file or topology buffer
- ▶ Standard MRNet calls to use created topology

Topology policy class members

- ▶ Number of internal processes per node
 - Default: 1
- ▶ Collocate internal processes with front-end
 - Default: Off
- ▶ Collocate internal processes with back-end
 - Default: Off
- ▶ Max fan out
 - Default: 128
- ▶ Min fan out
 - Default: 16
- ▶ Tree type: balanced, perfect, binomial, graded
 - Default: balanced
- ▶ Modes: “maximize fan out”, “maximize depth”, “fit to internal nodes”

Auto-topology to do list

- ▶ Performance testing of various topology types with representative tool workloads
 - Systematically chosen default policies
 - Inform use-case dependent policy customizations
- ▶ More robust testing and integration into mainline MRNet

General approach for MRNet extensions

- ▶ “External” add-on if possible
 - Minimize MRNet interface modifications
 - Minimize changes to MRNet base code
 - Can still package with MRNet
- ▶ Reduces barrier to acceptance
- ▶ Eases “blame attribution”

Potential Working Group Topics

- ▶ Autonomous tool operation in general
- ▶ Exascale tool readiness
 - Step 1: Change workshop name 😊
 - Will tool overhead increase with scale (like resilience)?
- ▶ How relevant are faults and failures for tools?
 - Maintain operation without recovering data?
 - Simple detection and restart?
 - More complex data recovery?

Questions

