UNM SCHOOL *of* ENGINEERING

Department of Computer Science

Scalable Tool Startup: Current and Upcoming LIBI Developments

Dorian Arnold
University of New Mexico

This Talk About …

Our recent and future work in:

- Overcoming the challenges of scalable tool startup

- This means whatever it takes to go
  - From a binary executable file on some storage medium

  - To running processes executing their primary functions

Scalable Startup Challenge #1: Launching the processes

- Launching $O(10^5)$ processes can take several minutes
- Insufficient responsiveness for:
  - Interactive tools:
    - Can take more time to deploy tool than to use it
  - Many-task computing and uncertainty quantification:
    - May launch many sets of short-lived tasks
- Existing resource managers use ad hoc strategies

Scalable Startup Challenge #2: Disseminating Initialization Information

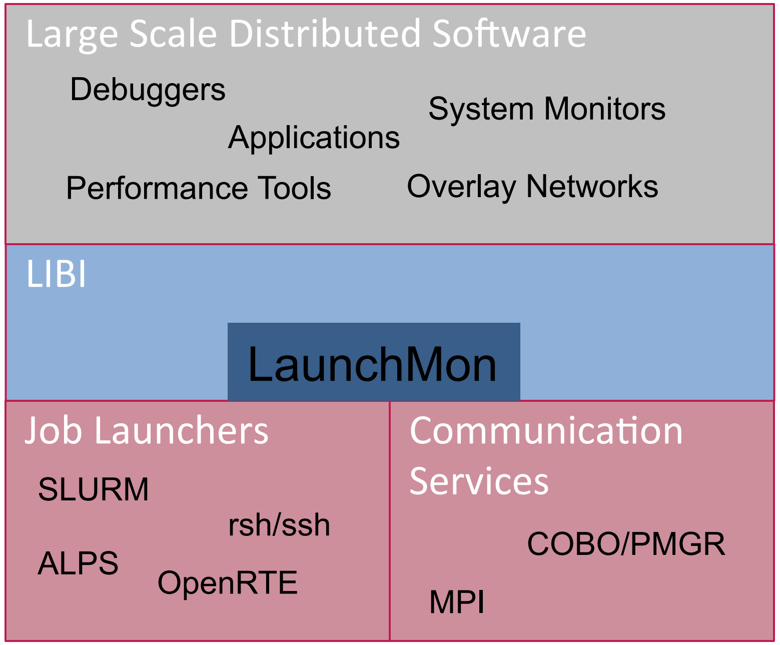
- Processes need startup information, e.g. initial configuration

- No standardized solutions
  - Each infrastructure uses its own custom mechanisms

- General problem: need a scalable infrastructure to launch a scalable infrastructure!

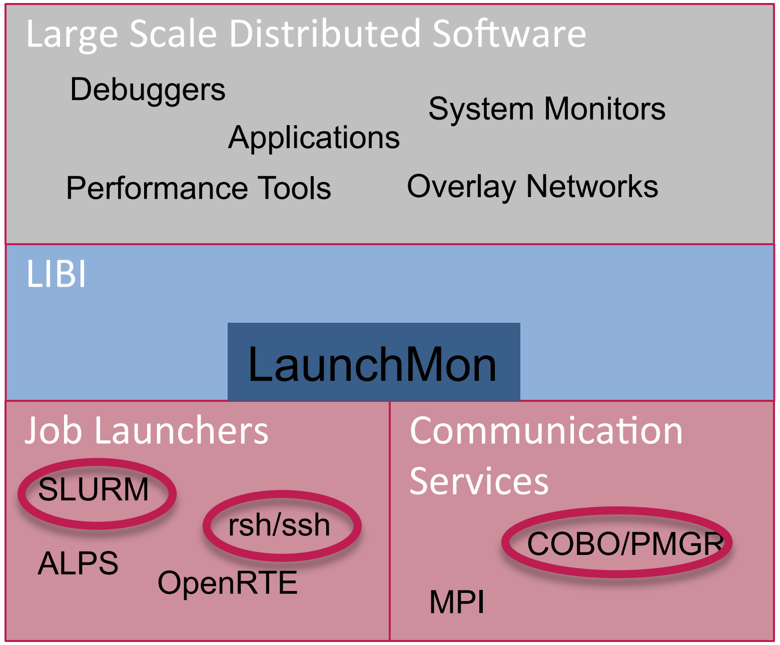
LIBI: The Lightweight Infrastructure-bootstrapping Infrastructure

- Assume native is best
  - Use native services when available;
  - Currently access native services via LaunchMon
- Be smart otherwise
  - Rsh-based
  - Customizable launch trees
    - Parents “rsh” children



Current LIBI

- Job launch
  - SLURM via LaunchMon
  - Rsh-based default
    - Optimal topology
- Information Dissemination
  - COBO via LaunchMon
  - Customizable launch topologies
  - Uses an optimal topology



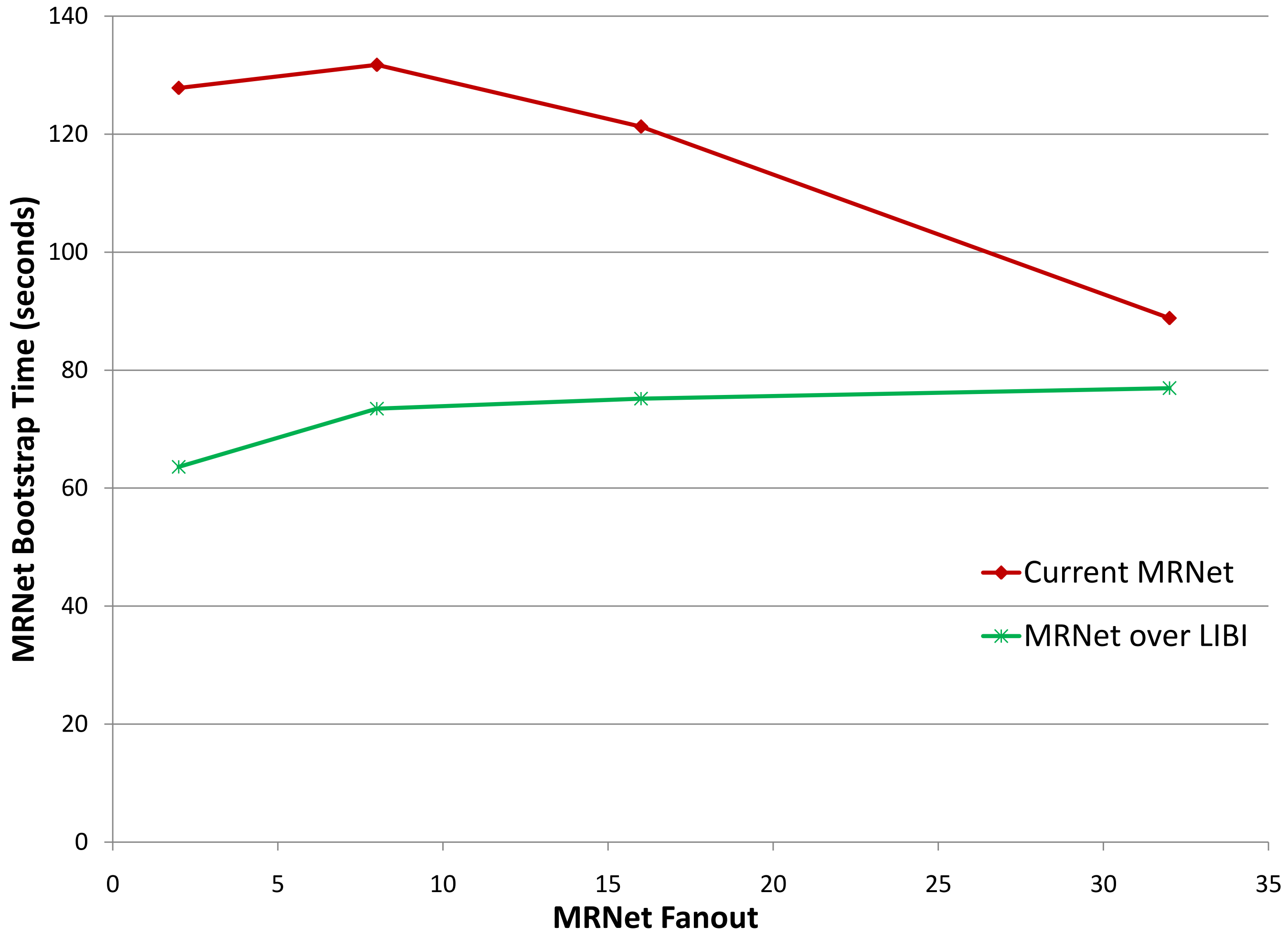
Launch Times for 1000 Processes

	SEQ=0.03, REM=0.172		SEQ=0.007,REM=2		SEQ=0.007, REM=10	
Rank	Topology	Launch (s)	Topology	Launch(s)	Topology	Launch (s)
1	Greedy	0.609	Greedy	4.272	Greedy	17.006
2	16	0.753	32	4.44	flat	17.006
3	32	0.784	64	4.552	32	20.44
4	8	0.841	128	4.944	64	20.552
5	64	0.896	256	5.812	128	20.944
6	4	0.971	16	6.237	256	21.812
7	128	1.288	512	7.422	512	23.422
8	2	1.624	8	8.153	16	30.237
9	256	2.156	flat	9.006	8	40.153
10	512	3.769	4	10.111	4	50.111
11	flat	7.178	2	18.076	2	90.076

Time to Find Optimal Tree

	100 tasks	1,000 tasks	10,000 tasks	100,000 tasks
16-ary tree	1.8E-4 secs	1.8E-3 secs	1.8E-2 secs	1.9E-1 secs
Optimal tree	2.5E-4 secs	2.1E-3 secs	2.3E-2 secs	2.8E-1 secs

MRNet Startup (varying MRNet fan-out)



LIBI/MRNet Integration Status

- LIBI integrated into MRNet
  - Currently a development branch in GIT
  - Tested on “vanilla” cluster
    - MRNet over LIBI over SLURM
    - MRNet over LIBI over ssh
  - MRNet uses non-LIBI mode where LIBI not ported
    - E.g., Cray XT*
  - Lightweight back-end mode not yet ported to LIBI
    - Challenge is C vs. C++ (and resources)

What We are Working on

Remember the two startup challenges:

1) Launching processes

1.5) Interconnecting launched processes

So processes can communicate with each other

2) Propagating initialization information

Scalable Startup Challenge #1.5: Interconnecting Processes

- To communicate, tasks must be (logically) interconnected

- How to bootstrap this interconnection?
  - Command line arguments
  - Environmental variables
  - File system
  - Information service

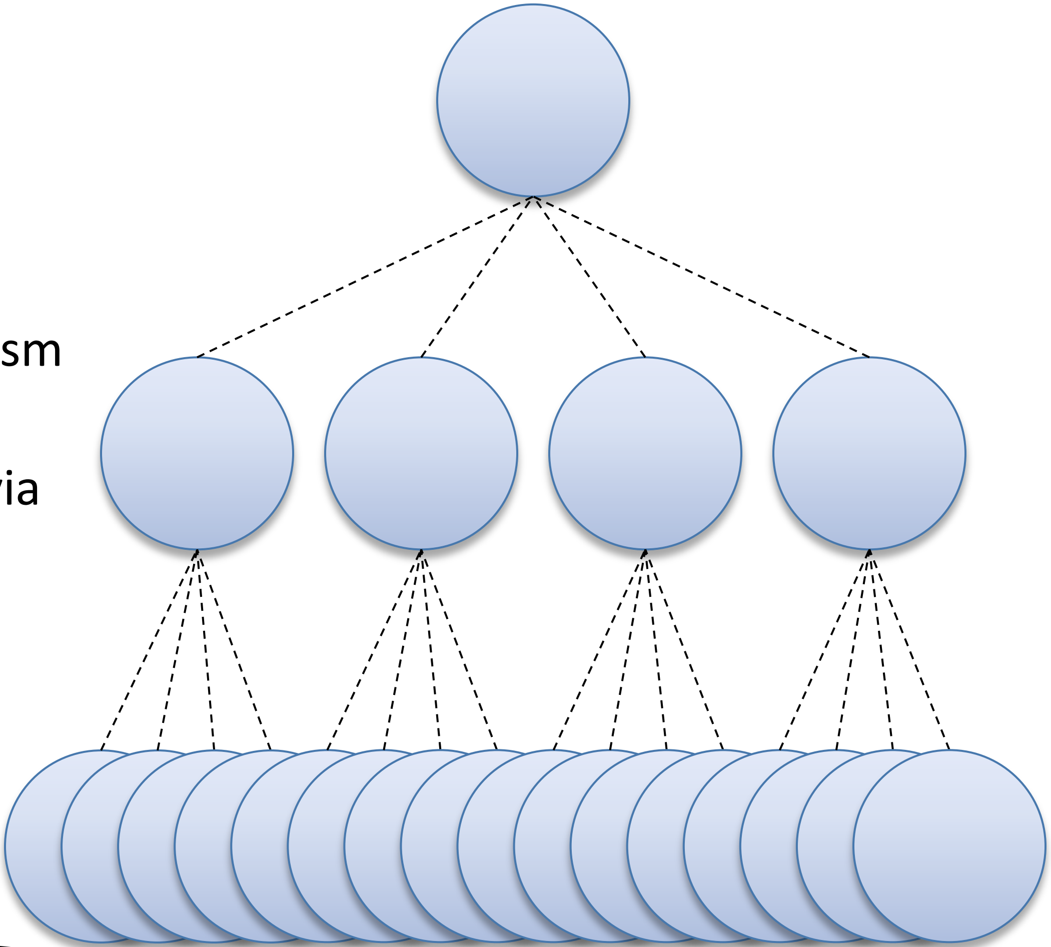
MRNet as the Use Case

- MRNet processes form tree to complete initialization

- Configuration information includes listening ports
  - Different per child (actually per set of children)
  - Dynamic: unavailable before parent initializes itself
  - Many (tens of thousands) of bits of small data
  - Access pattern: many publishers; many more subscribers

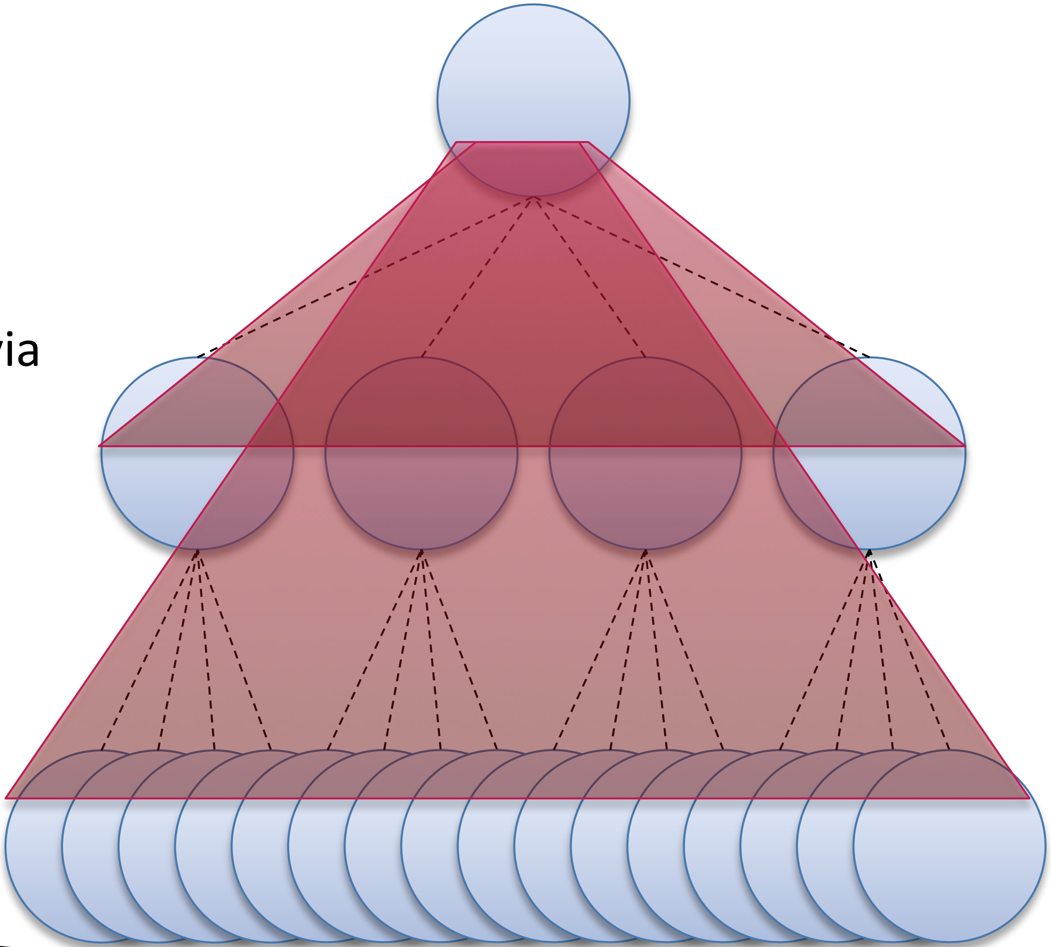
Different Startup Cases: Tree-based

- "Parent creates children"
  - Local ➔ `fork()/exec()`
  - Remote ➔ `rsh`-based mechanism
- Configuration information passed via command line or environment
  - Requires starting all processes



Different Startup Cases: Bulk

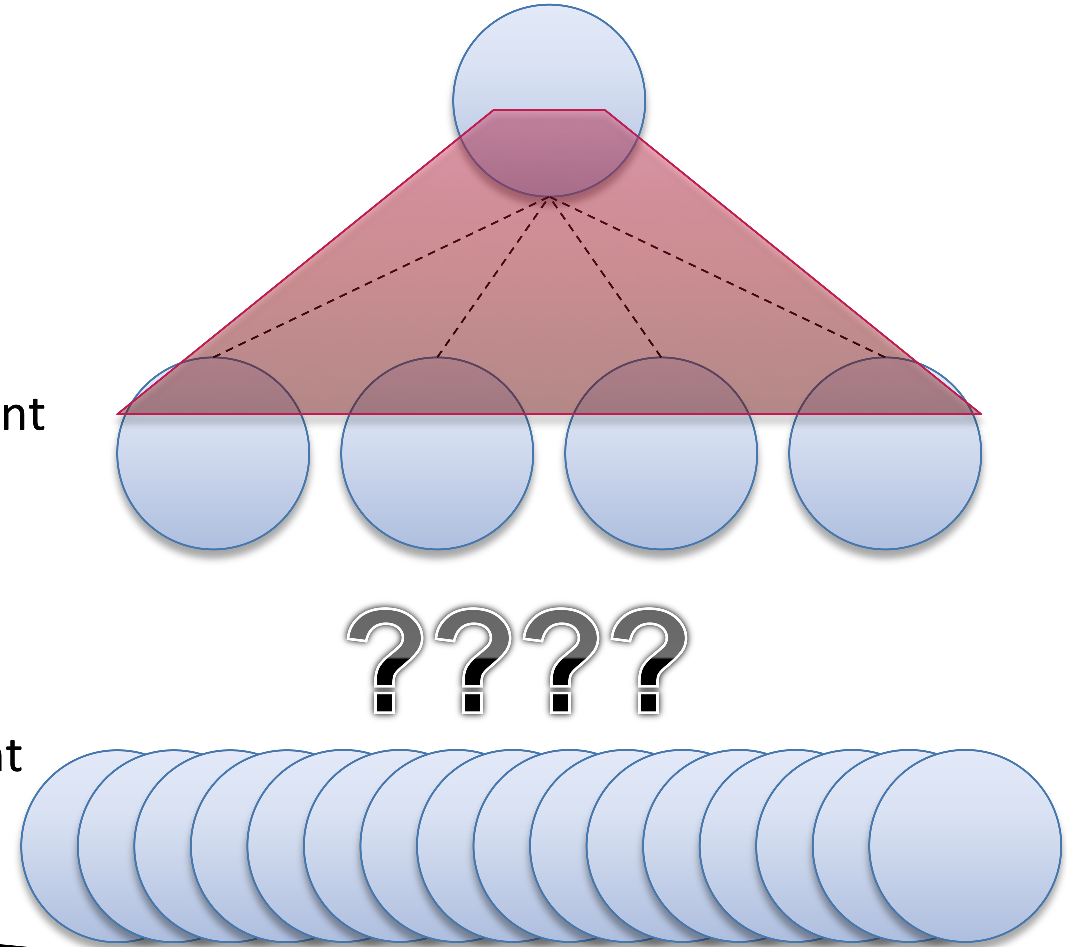
- "Root creates everyone"
  - LIBI, SLURM, etc.
- Configuration information passed via custom mechanism
  - PMGR-based collectives
  - Root gathers then scatters
- Requires starting all processes



Different Startup Cases: Disconnected Startup

Processes started at different times

- What do we do for "3rd party" processes?
  - Current solutions include inelegant ones like
    - Single file: one line per client
    - Multiple files: one file per client

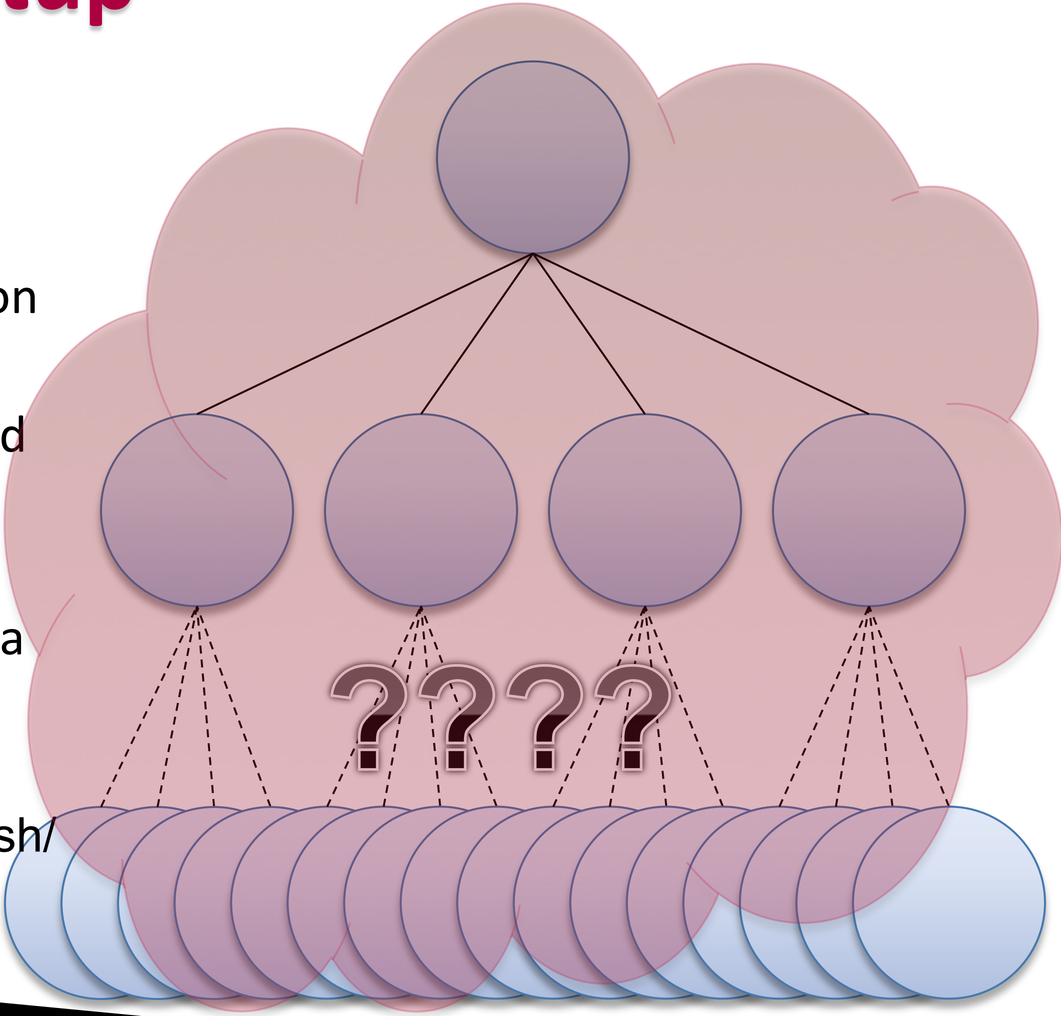


LIBI Extension

- A new “wire-up” protocol to inter-connect disconnected processes
- Part abstraction
  - How to properly encapsulate different connection mechanisms?
  - What minimal information is necessary?
- Part mechanism
  - Support different communication mechanisms:
  - What minimal level of persistence is necessary?
  - Still need some “hook”
    - Job ID, set of hosts, port range, etc.

Different Startup Cases: Disconnected Startup

- Processes started at different times
- All processes call `wire-up()` function
  - All processes inter-connected and ready for communication
- Configuration information passed via any number of generic mechanisms
  - E.g., group communication, publish/subscribe, key-value store



New LIBI “Wire-up” Status

- Working on first version of abstractions
- Target proof-of-concept implementation by end of summer
  - KVS-based
  - Leveraging PMI-based KVS and ZHT (a scalable KVS infrastructure from IIT)
- Integrate into MRNet

Other MRNet Updates: Requiring Less Topology Specification

- Currently: MRNet requires a complete topology specification
  - Must map every tree process to a host
  - Must specify how tree processes are inter-connected

- Goal: MRNet requires NO topology specification
  - No process/host mappings
  - No process inter-connections
  - No hosts
  - No nothing! ☺

A Step Toward Less Topology Specs

Things one can specify

- Physical hosts
  - Set of specific hosts
  - Number of hosts
  - Bounds on number of hosts
- MRNet processes
  - Number of back-ends
  - Placement of back-ends
  - Number of internal nodes
  - Placement of internal nodes
  - Processes per host bounds
- Process inter-connection
  - Specific topology
  - Tree shape (balanced, skewed, etc.)
  - Tree bounds (max fan-out, max depth, etc.)

New MRNet Startup Modes Status

- Proposal for new API extensions
  - Fully backward compatible
- Prototype of “set of hosts” mode
  - Where user gives hosts and MRNet autoconfigures tree
- End of summer target

When it comes to Job Startup …

The race is ~~not~~ to the swift
[and the battle to the scalable]

Questions for the Tools Experts

- Are we neglecting major impediments to large scale tool startup?
- Where do concepts like SPINDLE, SBRS and even CBTF fit in?
- What are your observed start up experiences?
- What would you like to see us working on?

Questions?