The Process Management Component of a Scalable Systems Software Environment

Rusty Lusk

(with Ralph Butler, Narayan Desai, Andrew Lusk)

Mathematics and Computer Science Division
Argonne National Laboratory
lusk@mcs.anl.gov

Outline

- Process management in general
- A component approach to systems software
  - The Scalable Systems Software Project
- Defining an abstract process management component
- A stand-alone process manager for scalable startup of MPI programs and other parallel jobs
  - MPD-2
  - An MPD-based implementation of the abstract definition
- Experiments and experiences with MPD and SSS software on a medium-sized cluster

Current State of Systems Software for Clusters

- Both proprietary and open-source systems
  - PBS, LSF, POE, SLURM, COOAE (Collections Of Odds And Ends), …
- Many are monolithic “resource management systems,” combining multiple functions
  - Job queuing, scheduling, process management, node monitoring, job monitoring, accounting, configuration management, etc.
- A few established separate components exist
  - Maui scheduler
  - Qbank accounting system
- Many home-grown, local pieces of software
- *Process Management* often a weak point

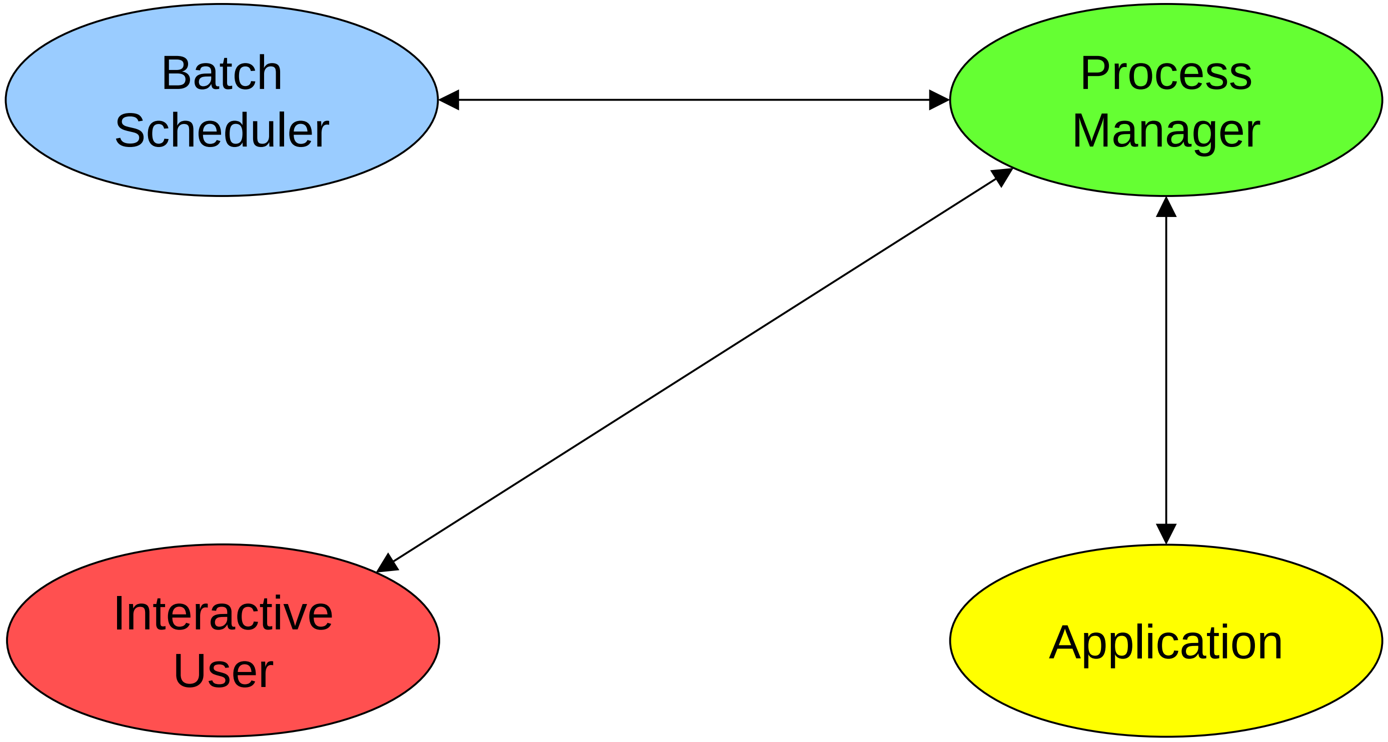
What is Process Management?

- A *process management system* is the software that starts user processes (with command line arguments and environment), ensures that they terminate cleanly, and manages I/O
- For simple jobs, this can be a normal Unix shell
- For parallel jobs, more is needed
- Process management is different from scheduling, queuing, and monitoring

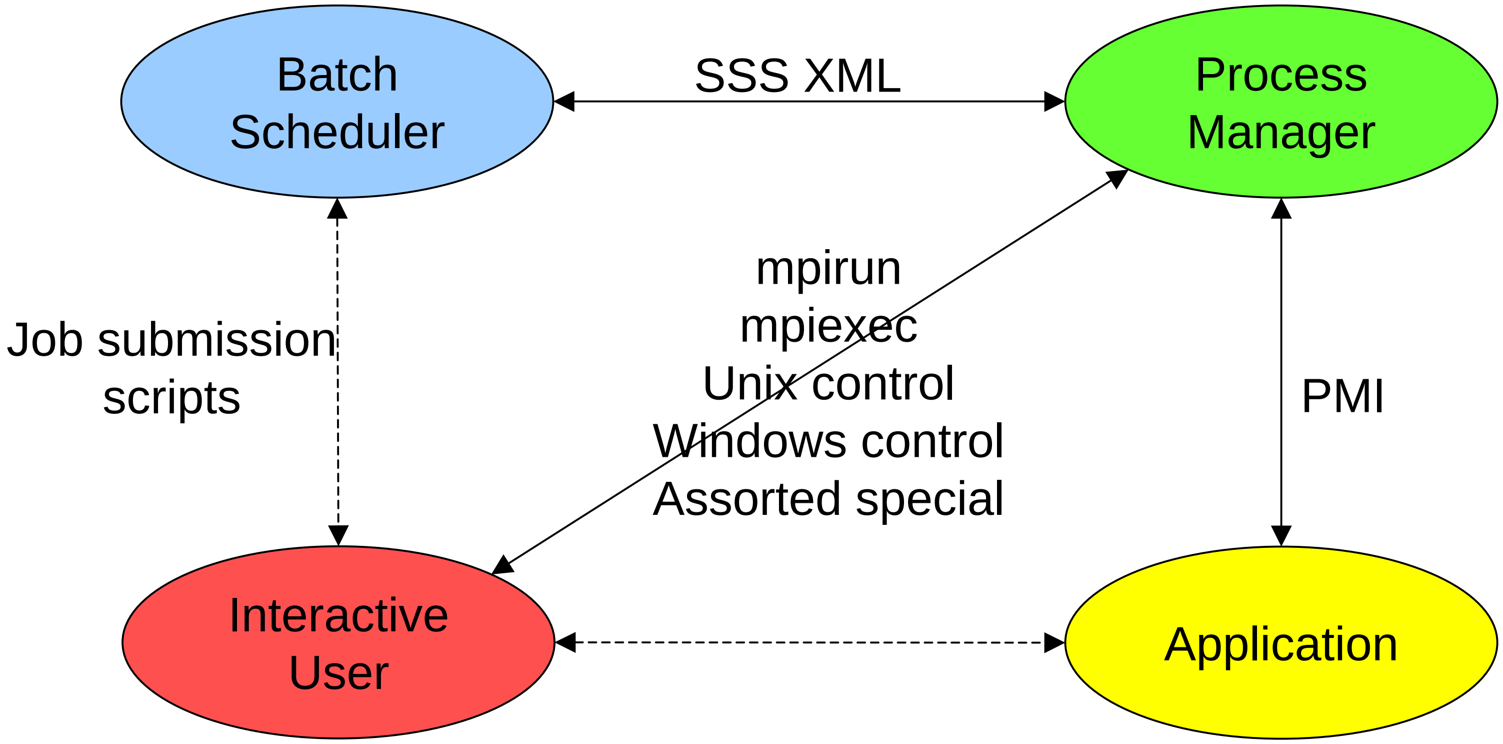
Typical Weaknesses of Process Managers

- Process startup not scalable
- Process startup not even parallel
  - May provide list of nodes and just start script on first one
  - Leaves application to do own process startup
- Parallel process startup may be restricted
  - Same executable, command-line arguments, environment
- Inflexible and/or non-scalable handling of stdin, stdout, stderr.
- Withholds useful information from parallel library
  - Doesn’t help parallel library processes find one another
  - No support for MPI-2 dynamic process management functions
- No particular support for tools
  - Debuggers, profilers, monitors
- And they are all different!

The Three “Users” of a Process Manager



Interfaces Are the Key



Process Manager Research Issues

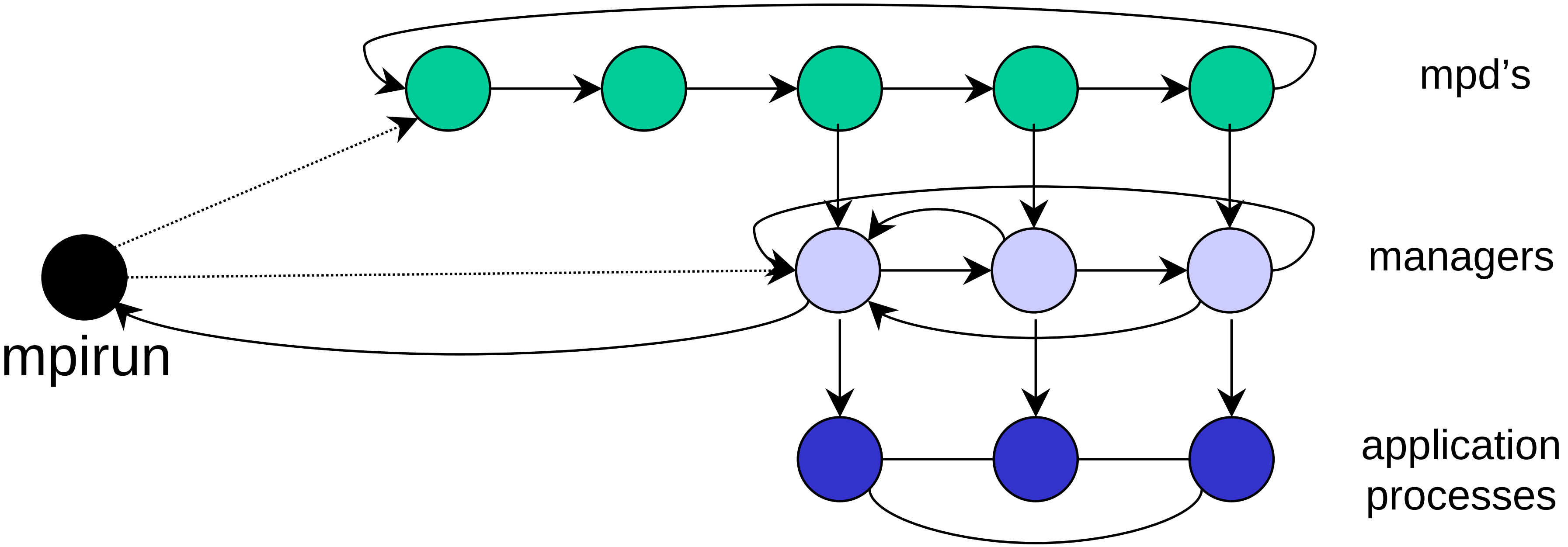
- Identification of proper process manager functions
  - Starting (with arguments and environment), terminating, signaling, handling stdio, …
- Interface between process manager and communication library
  - Process placement and rank assignment
  - Dynamic connection establishment
  - MPI-2 functionality: Spawn, Connect, Accept, Singleton Init
- Interface between process manager and rest of system software
  - Cannot be separated from system software architecture in general
  - Process manager is important component of component-based architecture for system software, communicating with multiple other components
- Scalability
  - A problem even on existing large systems
  - Some new systems coming present new challenges
  - Interactive jobs (such as Scalable Unix Tools) need to start fast

Background – The MPD Process Manager

- Primary research goals:
  - Fast and scalable startup of parallel jobs (especially MPICH)
  - Explore interface needed to support MPI and other parallel libraries
    - Helping processes locate and connect to other processes in job, in scalable way (the BNR interface)
- Part of MPICH-1
  - ch_p4mpd device
- Established that MPI job startup could be very fast
  - Encouraged interactive parallel jobs
  - Allowed some system programs (e.g. file staging) to be written as MPI programs (See Scalable Unix Tools, EuroPVM/MPI-8)

MPD-1

Architecture of MPD:



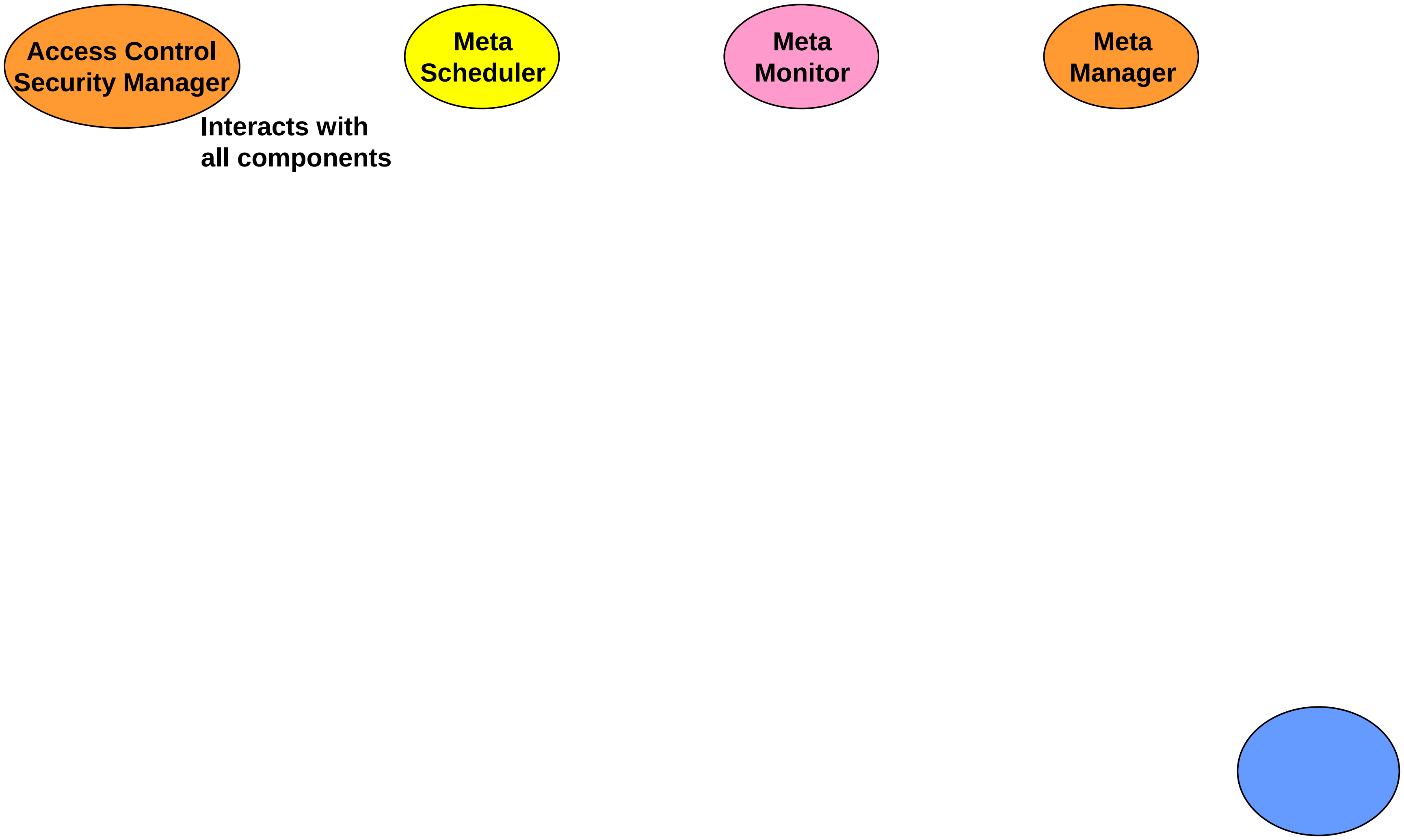
Recent Developments

- Clusters get bigger, providing a greater need for scalability
- Large clusters serve many users
  - Many issues the same for “non-cluster” machines
- MPI-2 functionality puts new demands on process manager
  - MPI_Comm_spawn
  - MPI_Comm_connect, MPI_Comm_accept, MPI_Comm_join
- MPICH-2 provides opportunity to redesign library/process manager interface
- Scalable Systems Software SciDAC project presents an opportunity to consider Process Manager as a separate component participating in a component-based systems software architecture
- New requirements for systems software on research cluster at Argonne

The Scalable Systems Software SciDAC Project

- Multiple Institutions (most U. S. national labs, plus NCSA)
- Research goal: to develop a component-based architecture for systems software for scalable machines
- Software goal: to demonstrate this architecture with some prototype open-source components
- One powerful effect: forcing rigorous (and aggressive) definition of what a process manager should do and what should be encapsulated in other components
- http://www.scidac.org//ScalableSystems

System Software Components



Access Control Security Manager

Interacts with all components

Meta Scheduler

Meta Monitor

Meta Manager

Defining Process Management in the Abstract

- Define functionality of process manager component
- Define interfaces by which other components can invoke process management services
- Try to avoid specifying how system will be managed as a whole
- Start by deciding what should be included and not included

Not Included

- Scheduling
  - Another component will either make scheduling decisions (selection of hosts, time to run), or explicitly leave host selection up to process manager
- Queueing
  - A job scheduled to run in the future will be maintained by another component; the process manager will start jobs immediately
- Node monitoring
  - The state of a node is of interest to the scheduler, which can find this out from another component
- Process monitoring
  - CPU usage, memory footprint, etc, are attributes of individual processes, and can be monitored by another component. The process manager can help by providing job information (hosts, pids)
- Checkpointing
  - Process manager can help with signals, but CP is not its job

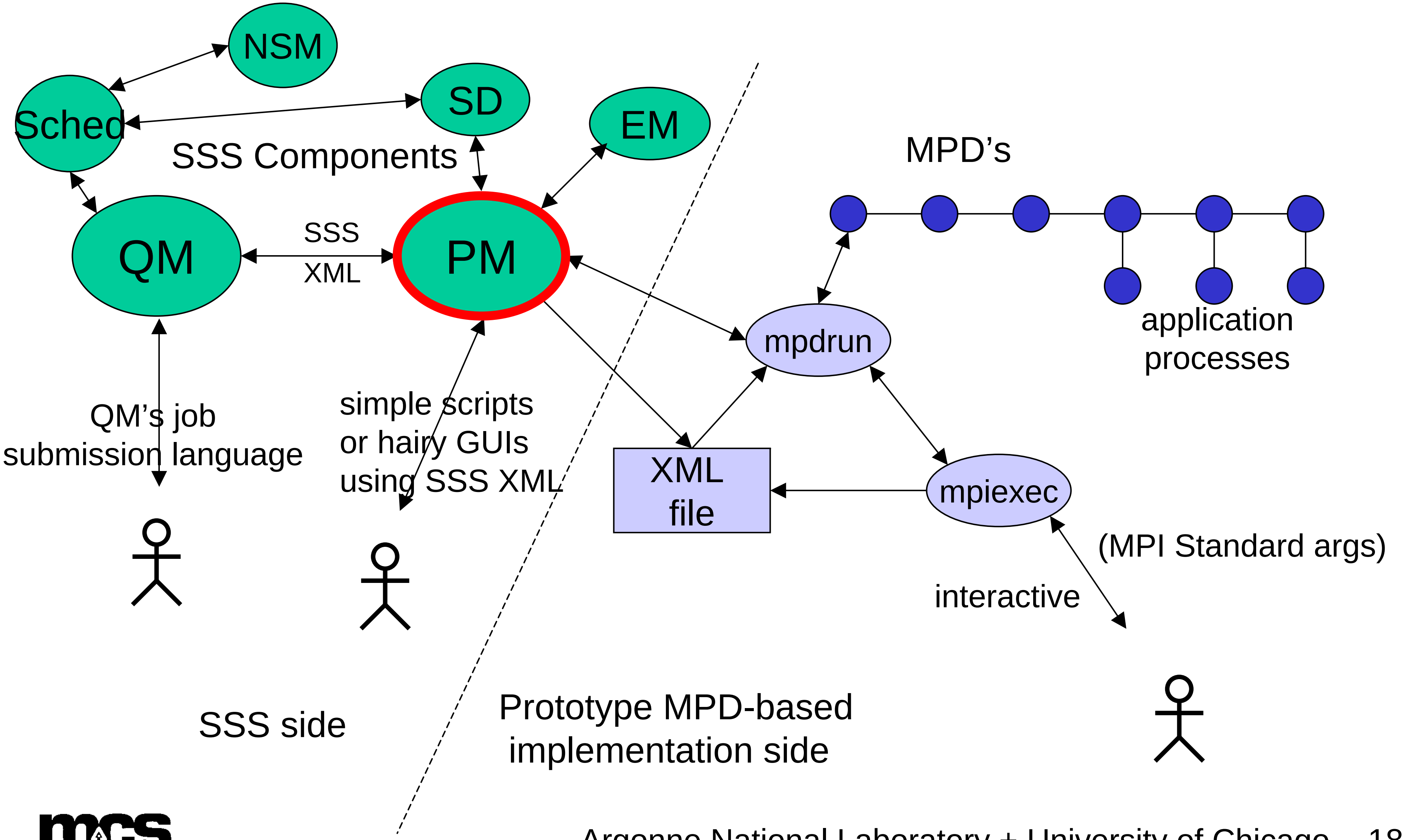
Included

- Starting a parallel job
  - Can specify multiple executables, arguments, environments
- Handling stdio
  - Many options
- Starting co-processes
  - Tools such as debuggers and monitors
- Signaling a parallel job
- Killing a parallel job
- Reporting details of a parallel job
- Servicing the parallel job
  - Support MPI implementation, other services
- In context of Scalable Systems Software suite, register so that other components can find it, and report events

The SSS Process Manager

- Provides previously-listed functions
- Communicates with other SSS components using XML messages over sockets (like other SSS components do)
- Defines syntax and semantics of specific messages:
  - Register with service directory
  - Report events like job start and termination
  - Start job
  - Return information on a job
  - Signal job
  - Kill job
- Uses MPD-2 to carry out its functions

Schematic of Process Management Component in Scalable Systems Software Context



MPD-2: Second-Generation MPD

- Same basic architecture as MPD-1
- Provides new functionality required by SSS definition
  - E.g., separate environment variables for separate ranks
- Provides new interface for parallel library like MPICH-2
  - PMI interface extends, improves, generalizes previous interface
    - Multiple key-val spaces
    - Put/get/fence interface for scalability
    - Spawn/accept/connect at low level to support MPI-2 functions
- Maintains scalability features of MPD
- Improved fault-tolerance
- Can run as root and start multiple jobs for multiple users

Testing the MPD Ring

- Here the ring of MPD’s had 206 hosts
- Simulated larger ring by sending message around ring multiple times

Times around the ring	Time in seconds
1	.13
10	.89
100	8.93
1000	89.44

- Linear, as expected
- But fast: > 2000 hops/sec

Running Non-MPI Jobs

- Ran **`hostname`** on each node
- Creates stdio tree and collects output from each node
- Sublinear

Number of hosts	Time in seconds
1	.83
4	.86
8	.92
16	1.06
32	1.33
64	1.80
128	2.71
192	3.78

Running MPI Jobs

- Ran cpi on each node (includes I/O, Bcast, Reduce)
- Compared MPICH-1 (ch_p4 device) with MPICH-2 with MPD-2

Number of Processes	Old Time	New Time
1	.4	.63
4	5.6	.67
8	14.4	.73
16	30.9	.86
32	96.9	1.01
64		1.90
128		3.50

- Better!

SSS Project Global Issues

- Put minimal constraints on component implementations
  - Ease merging of existing components into SSS framework
    - E.g., Maui scheduler
  - Ease development of new components
  - Encourage multiple implementations from vendors, others
- Define minimal global structure
  - Components need to find one another
  - Need common communication method
  - Need common data format at some level
    - Each component will compose messages others will read and parse
  - Multiple message-framing protocols allowed

SSS Project Status – Global

- Early decisions on inter-component communication
  - Lowest level communication is over sockets (at least)
  - Message content will be XML
    - Parsers available in all languages
  - Did not reach consensus on transport protocol (HTTP, SOAP, BEEP, assorted home grown), especially to cope with local security requirements
- Early implementation work on global issues
  - Service directory component defined and implemented
  - SSSlib library for inter-component communication
    - Handles interaction with SD
    - Hides details of transport protocols from component logic
    - Anyone can add protocols to the library
    - Bindings for C, C++, Java, Perl, and Python

Chiba City

- Medium-sized cluster at Argonne National Laboratory
  - 256 dual-processor 500MHz PIII's
  - Myrinet
  - Linux (and sometimes others)
  - No shared file system, for scalability
- Dedicated to Computer Science scalability research, not applications
- Many groups use it as a research platform
  - Both academic and commercial
- Also used by friendly, hungry applications
- New requirement: support research requiring specialized kernels and alternate operating systems, for OS scalability research

New Challenges

- Want to schedule jobs that require node rebuilds (for new OS's, kernel module tests, etc.) as part of "normal" job scheduling
- Want to build larger virtual clusters (using VMware or User Mode Linux) temporarily, as part of "normal" job scheduling
- Requires major upgrade of Chiba City systems software

Chiba Commits to SSS

- Fork in the road:
  - Major overhaul of old, crufty, Chiba systems software (open PBS + Maui scheduler + homegrown stuff), OR
  - Take leap forward and bet on all-new software architecture of SSS
- Problems with leaping approach:
  - SSS interfaces not finalized
  - Some components don’t yet use library (implement own protocols in open code, not encapsulated in library)
  - Some components not fully functional yet
- Solutions to problems:
  - Collect components that are adequately functional and integrated (PM, SD, EM, BCM)
  - Write “stubs” for other critical components (Sched, QM)
  - Do without some components (CKPT, monitors, accounting) for the time being

Features of Adopted Solution

- Stubs quite adequate, at least for time being
  - Scheduler does FIFO + reservations + backfill, improving
  - QM implements “PBS compatibility mode” (accepts user PBS scripts) as well as asking Process Manager to start parallel jobs directly
- Process Manager wraps MPD-2, as described above
  - Single ring of MPD’s runs as root, managing all jobs for all users
  - MPD’s started by Build-and-Config manager at boot time
- An MPI program called MPISH (MPI Shell) wraps user jobs for handling file staging and multiple job steps
- Python implementation of most components
- Demonstrated feasibility of using SSS component approach to systems software
  - Running normal Chiba job mix for over a month now
  - Moving forward on meeting new requirements for research support

Summary

- Scalable process management is a challenging problem, even just from the point of view of starting MPI jobs
- Designing an abstract process management component as part of a complete system software architecture helped refine the precise scope of process management
- Original MPD design was adopted to provide core functionality of an SSS process manager without giving up independence (can still start MPI jobs with **`mpiexec`**, without using SSS environment)
- This Process Manager, together with other SSS components, has demonstrated the feasibility and usefulness of a component-based approach to advanced systems software for clusters and other parallel machines.

The End