HPC Controls: View to the Future

Ralph H. Castain
June, 2015

This Presentation: A Caveat

- One person's view into the future
  - Focusing on the open source community
  - Compiled from presentations and emails with that community, the national labs, various corporations, etc.
  - Spans last 20+ years
  - Not indicative of what any corporation or organization is doing or may do in the future

Overview

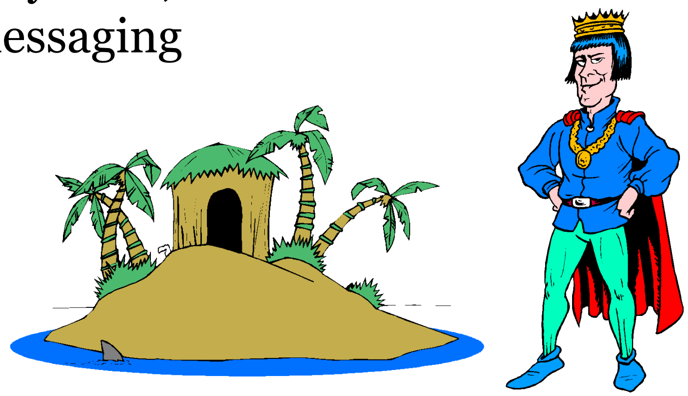
- Resource Manager
  - Workload manager
  - Run-time environment
- Monitoring
- Resiliency/Error Management
- Overlay Network
- Pub-sub Network
- Console

Controls Overview: A Definition

- Resource Manager
  - Workload manager
  - Run-time environment
- Monitoring
- Resiliency/Error Management
- Overlay Network
- Pub-sub Network
- Console

Open Ecosystems Today

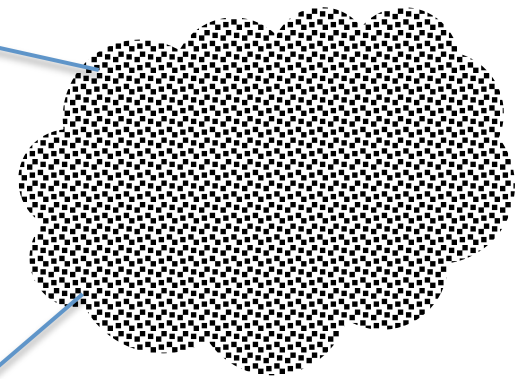
- Every element is independent island, each with its own community
- Multiple programming languages
- Conflicting licensing
- Cross-element interactions, where they exist, are via text-based messaging



Controls Overview: A Definition

- Resource Manager
  - Workload manager
  - Run-time environment
- Monitoring
- Resiliency/Error Management
- Overlay Network
- Pub-sub Network
- Console

Future Approach



- Common software environment
  - Pool of plugins that can be assembled into different tools
- Leverage external communities

General Requirements

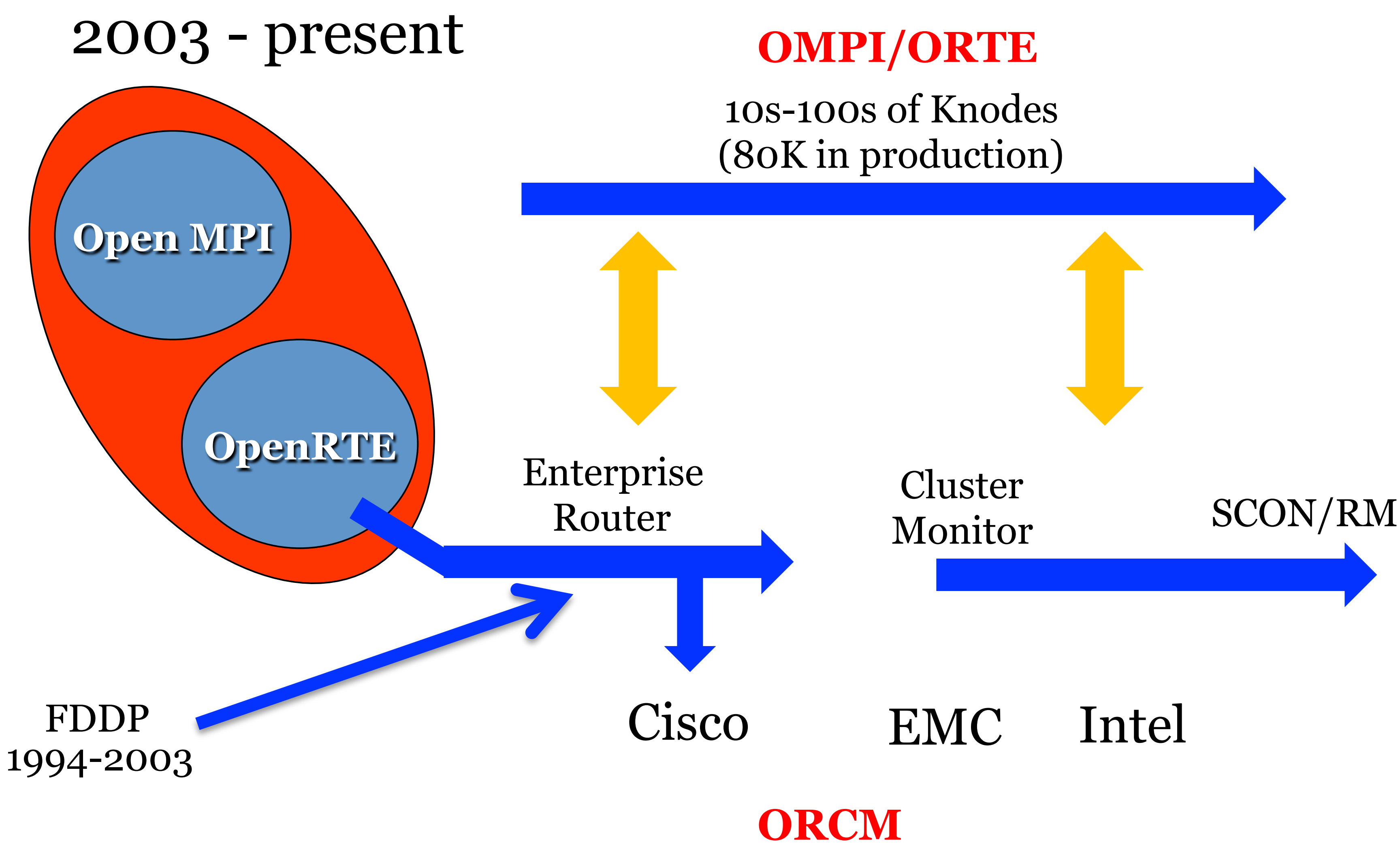
- Scalable to exascale levels & beyond
  - Better-than-linear scaling
  - Constrained memory footprint
- Dynamically configurable
  - Sense and adapt, user-directable
  - On-the-fly updates
- Open source (non-GPL)*
- Maintainable, flexible
  - Single platform that can be utilized to build multiple tools
  - Existing ecosystem
- Resilient
  - Self-heal around failures
  - Reintegrate recovered resources

*allow cross-integration without license conflicts

Reference Software Platform

- Demonstrated scalability
- Established community
- Clean non-GPL licensing
- Compatible architecture
- …

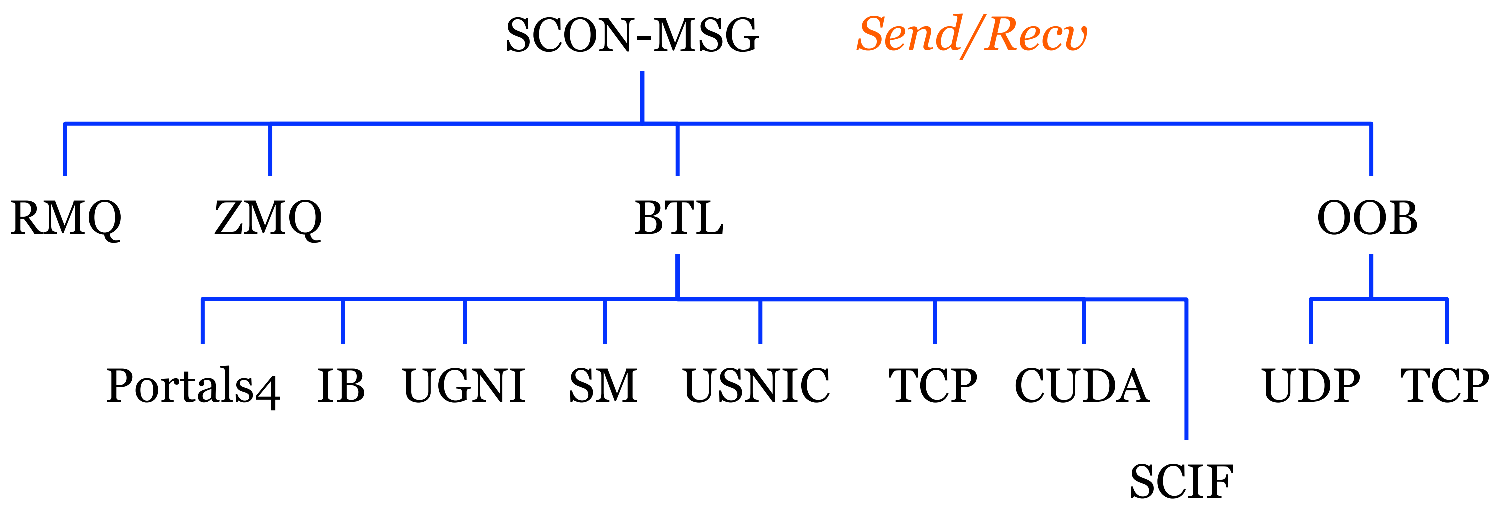
Open Resilient Cluster Manager (ORCM)


2003 - present
OMPI/ORTE
10s-100s of Knodes
(80K in production)
Open MPI
OpenRTE
Enterprise
Router
Cluster
Monitor
SCON/RM
FDDP
1994-2003
Cisco
EMC
Intel
ORCM

Abstractions

- Divide functional blocks into abstract frameworks
  - Standardize the API for each identified functional area
  - Can be used external, or dedicated for internal use
  - Single-select: pick active component at startup
  - Multi-select: dynamically decide for each execution
- Multiple implementations
  - Each in its own isolated plugin
  - Fully implement the API
  - Base “component” holds common functions

Example: SCalable Overlay Network (SCON)



ORCM and Plug-ins

- Plug-ins are shared libraries
  - Central set of plug-ins in installation tree
  - Users can also have plug-ins under $HOME
  - Proprietary binary plugins picked up at runtime
- Can add / remove plug-ins after install
  - No need to recompile / re-link apps
  - Download / install new plug-ins
  - Develop new plug-ins safely
- Update “on-the-fly”
  - Add, update plug-ins while running
  - Frameworks “pause” during update

Controls Overview: A Definition

- Resource Manager
  - Workload manager
  - Run-time environment
- Monitoring
- Resiliency/Error Management
- Overlay Network
- Pub-sub Network
- Console

Definition: RM

- Scheduler/Workload Manager
  - Allocates resources to session
  - Interactive and batch
- Run-Time Environment
  - Launch and monitor applications
  - Support inter-process communication wireup
  - Serve as intermediary between applications and WM
    - Dynamic resource requests
    - Error notification
  - Implement failure policies

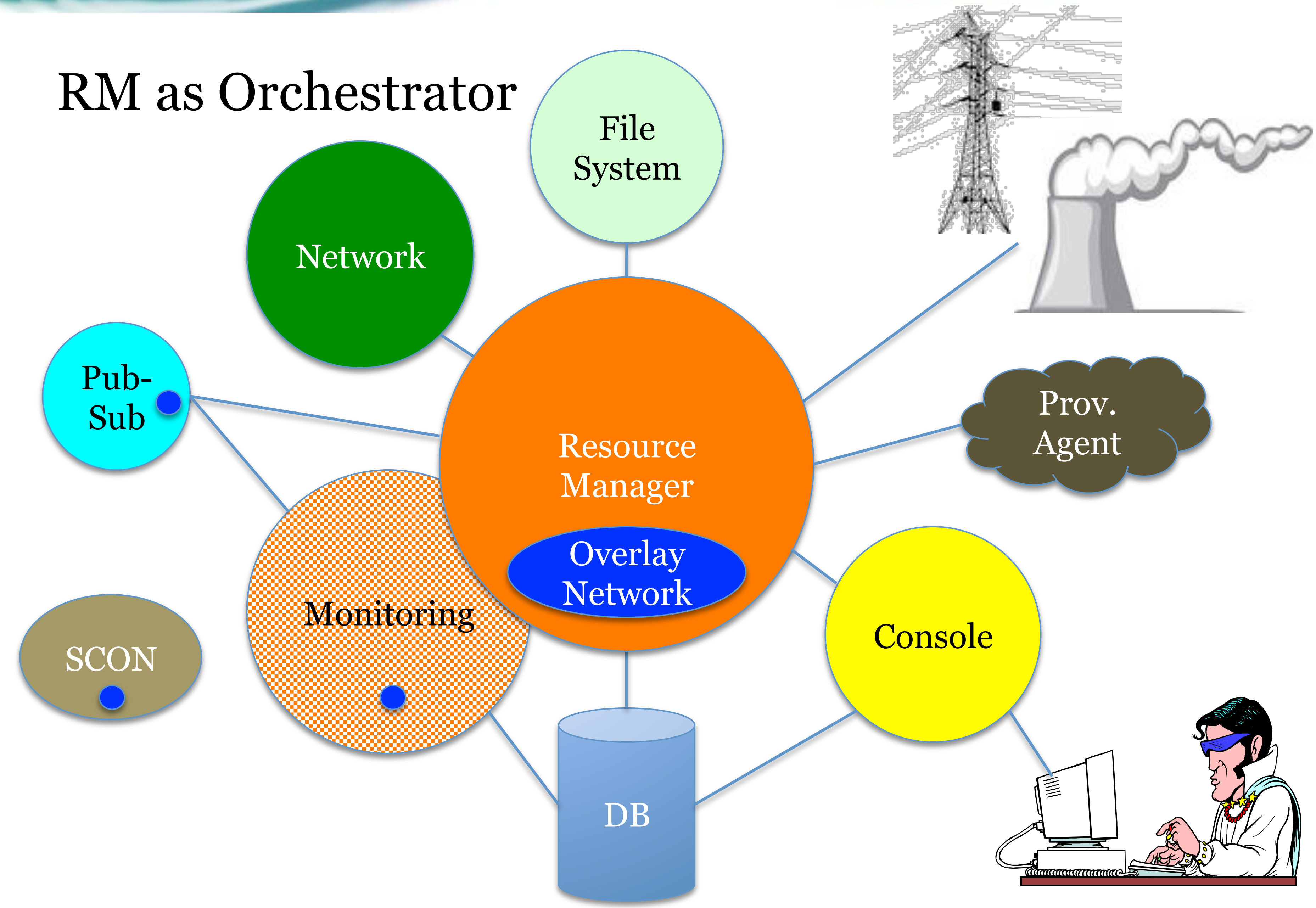
Breaking it Down

- Workload Manager
  - Dedicated framework
  - Plugins for two-way integration to external WM (Moab, Cobalt)
  - Plugins for implementing internal WM (FIFO)
- Run-Time Environment
  - Broken down into functional blocks, each with own framework
    - Loosely divided into three general categories
    - Messaging, launch, error handling
    - One or more frameworks for each category
  - Knitted together via “state machine”
    - Event-driven, async
    - Each functional block can be separate thread
    - Each plugin within each block can be separate thread(s)

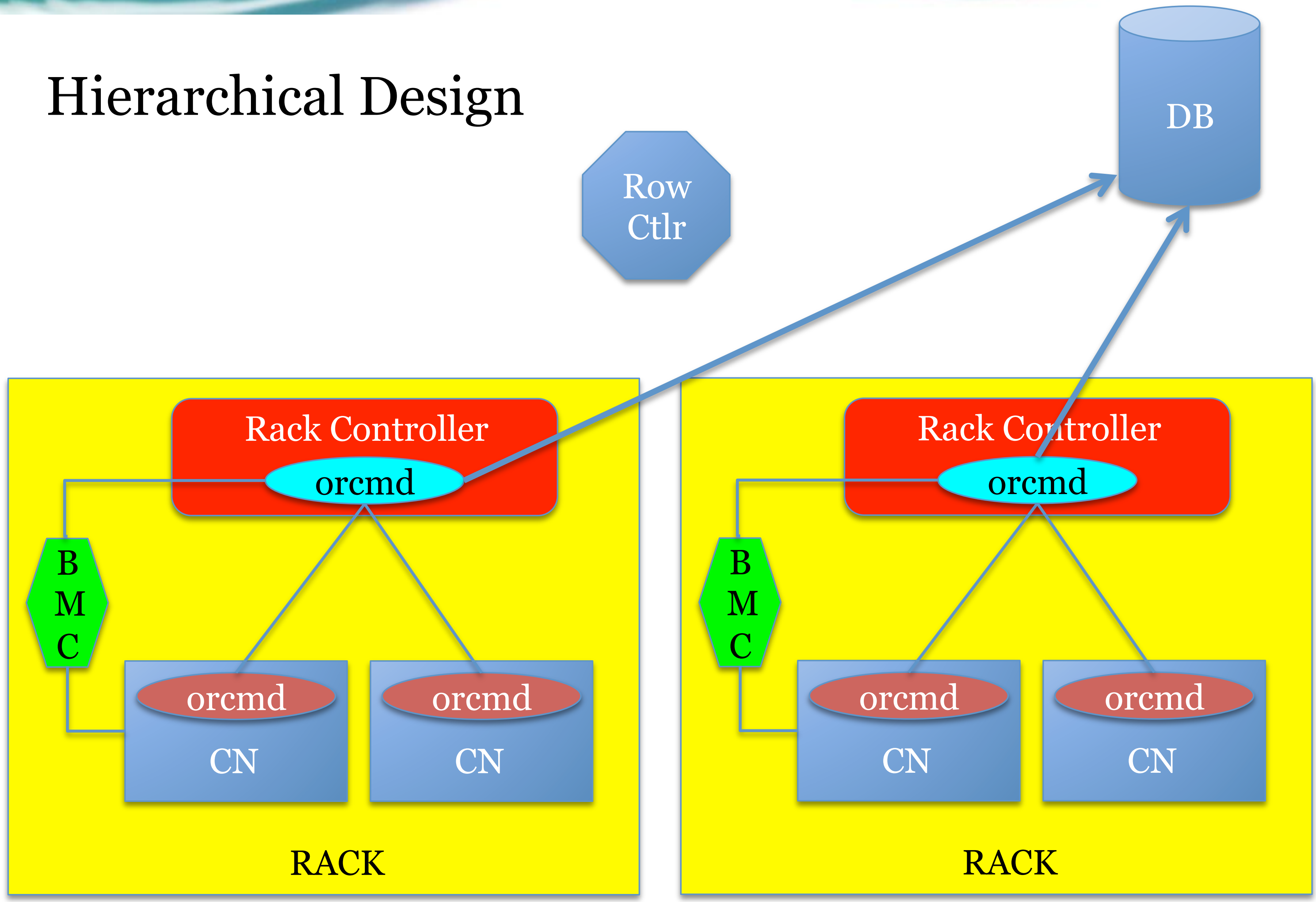
Key Objectives for Future

- Orchestration via integration
  - Monitoring system, file system, network, facility, console
- Power control/management
- “Instant On”
- Application interface
  - Request RM actions
  - Receive RM notifications
  - Fault tolerance
- Advanced workload management algorithms
  - Alternative programming model support (Hadoop, Cloud)
  - Anticipatory scheduling

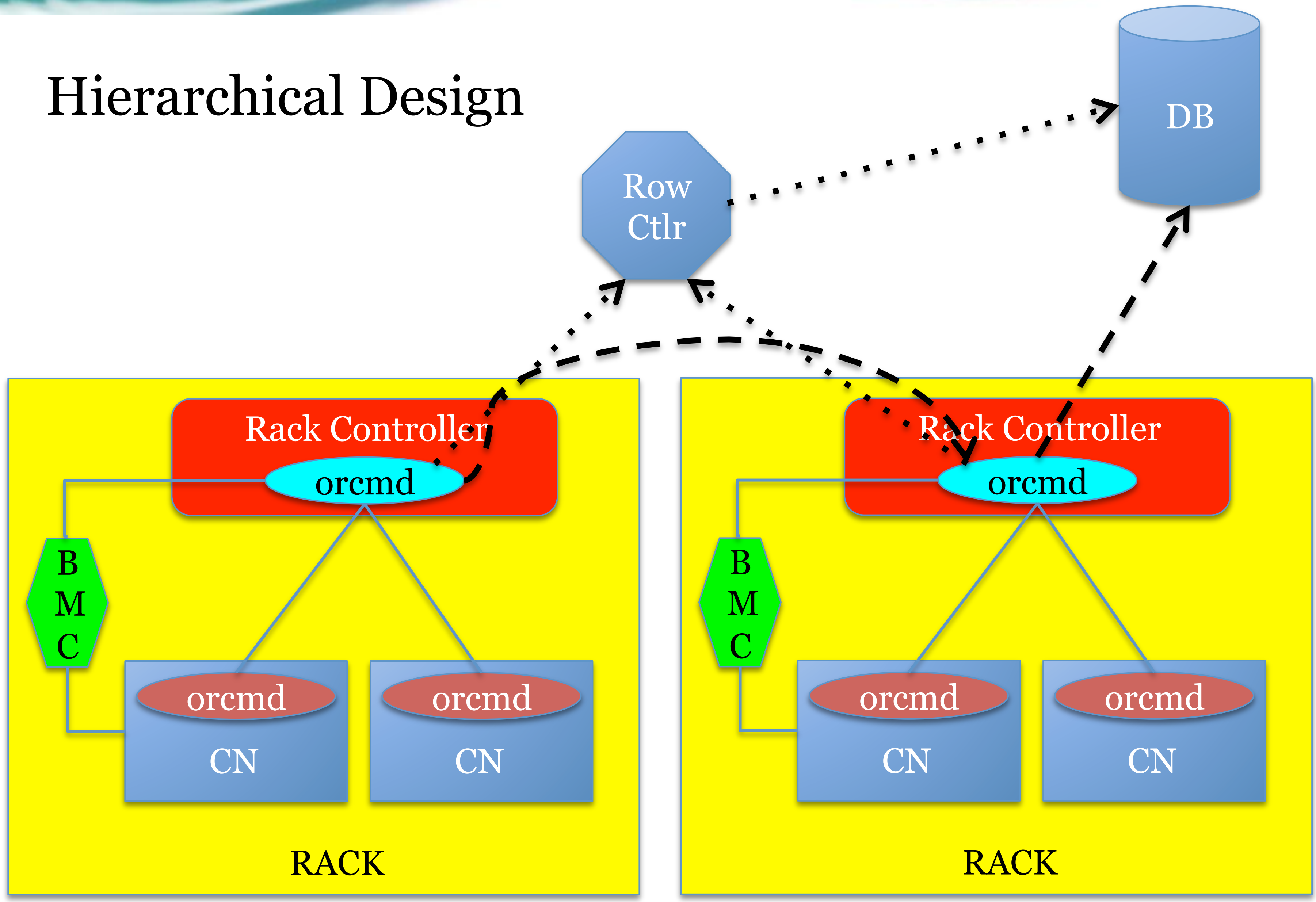
RM as Orchestrator



Hierarchical Design



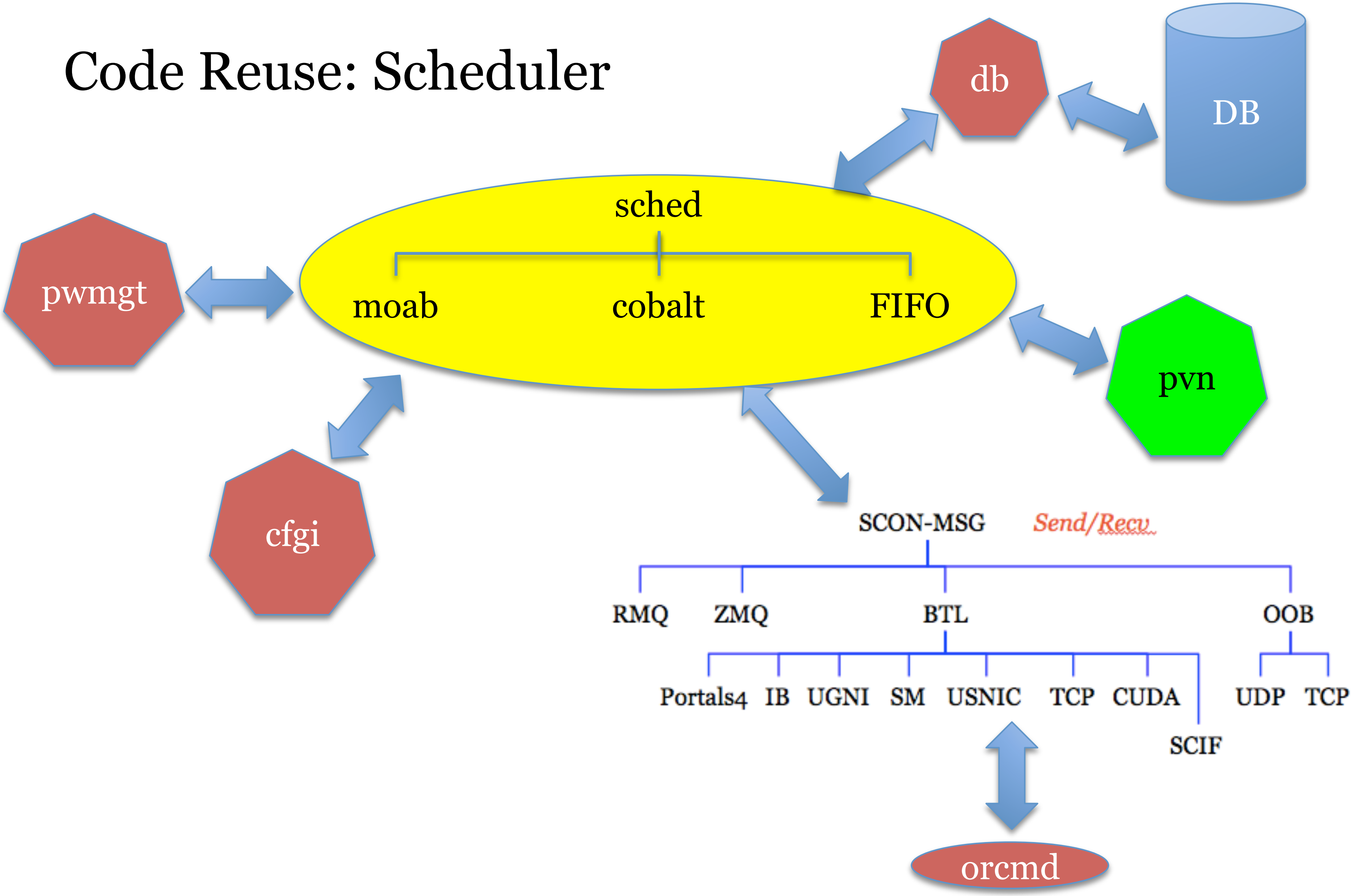
Hierarchical Design



Flexible Architecture

- Each tool built on top of same plugin system
  - Different combinations of frameworks
  - Different plugins activated to play different roles
  - Example: orcmd on compute node vs on rack/row controllers
- Designed for distributed, centralized, hybrid operations
  - Centralized for small clusters
  - Hybrid for larger clusters
  - Example: centralized scheduler, distributed “worker-bees”
- Accessible to users for interacting with RM
  - Add shim libraries (abstract, public APIs) to access framework APIs
  - Examples: SCON, pub-sub, in-flight analytics

Code Reuse: Scheduler



File System Integration

- Input
  - Current data location, time to retrieve and position
  - Data the application intends to use
    - Job submission, dynamic request (via RTE), persistence across jobs and sessions
  - What OS/libraries application requires
- Workload Manager
  - Scheduling algorithm factors
    - Current provisioning map, NVM usage patterns/requests, persistent data/ckpt location
    - Data/library locality, loading time
- Output
  - Pre-location requirements to file system (e.g., SPINDLE cache tree)
  - Hot/warm/cold data movement, persistence directives to RTE
  - NVM & Burst Buffer allocations (job submission, dynamic)

Provisioning Integration

- Support multiple provisioning agents
  - Warewulf, xCAT, ROCKS
- Support multiple provisioning modes
  - Bare metal, Virtual Machines (VMs)
- Job submission includes desired provisioning
  - Scheduler can include provisioning time in allocation decision, anticipate re-use of provisioned nodes
  - Scheduler notifies provisioning agent of what nodes to provision
  - Provisioning agent notifies RTE when provisioning complete so launch can proceed

Network Integration

- Quality of service allocations
  - Bandwidth, traffic priority, power constraints
  - Specified at job submission, dynamic request (via RTE)
- Security requirements
  - Network domain definitions
- State-of-health
  - Monitored by monitoring system
  - Reported to RM for response
- Static endpoint
  - Allocate application endpoints prior to launch
    - Minimize startup time
  - Update process location upon fault recovery

Power Control/Management

- Power/heat-aware scheduling
  - Specified cluster-level power cap, ramp up/down rate limits
  - Specified thermal limits (ramp up/down rates, level)
  - Node-level idle power, shutdown policies between sessions/time-of-day
- Site-level coordination
  - Heat and power management subsystem
    - Consider system capacity in scheduling
    - Provide load anticipation levels to site controllers
  - Coordinate ramps (job launch/shutdown)
    - Within cluster, across site
  - Receive limit (high/low) updates
  - Direct RTE to adjust controls while maintaining sync across cluster

"Instant On"

- Objective
  - Reduce startup from minutes to seconds
  - 1M procs, 50k nodes thru MPI_Init
    - On track: 20 sec
    - 2018 target: 5 sec
- What we require
  - Use HSN for communication during launch
  - Static endpoint allocation prior to launch
  - Integrate file system with RM for prepositioning, with scheduler for anticipatory locations

“Instant On” Value-Add

- Requires two things
  - Process can compute endpoint of any remote process
  - Hardware can reserve and translate virtual pids (vpid) to local pid
- RM knows process map
  - HW team: provide vpid-to-pid lookup tables, execute lookup
  - Fox River team: define driver hook for creating local lookup table
  - OFI team: define API for abstracted driver hook
  - MPI team: get process map from RM, translate to host:vpid for messages
- Eliminates costly sharing of endpoint info
- Significantly impact fault tolerance model

Application Interface: PMIx

- Current PMI implementations are limited
  - Only used for MPI wireup
  - Don’t scale adequately
    - Communication required for every piece of data
    - All blocking operations
  - Licensing and desire for standalone client library
- Increasing requests for app-RM interactions
  - Job spawn, data pre-location, power control
  - Current approach is fragmented
    - Every RM creating its own APIs

Application Interface: PMIx

- Current PMI implementations are limit
  - Only used for MPI wireup
  - Don’t scale adequately
    - Communication requi r every piece of data
    - All blocking operat
  - Licensing and desire fo dalone client library
- Increasing request pp-RM interactions
  - Job spawn, dat location, power control
  - Current approach is fragmented
    - Every RM creating its own APIs

Solve both with one solution!

PMIx Approach

- Ease adoption
  - Backwards compatible to PMI-1/2 APIs
  - Standalone client library
  - Server convenience library

  First time
- BSD-licensed
- Subproject of OMPI to utilize existing community
- Replace blocking with non-blocking operations for scalability
- Add APIs to support
  - New use-cases: IO, power, error notification, checkpoint, …
  - Programming models beyond MPI

PMIx: Fault Tolerance

- Notification
  - App can register for error notifications, incipient faults
  - RM will notify when app would be impacted
    - Notify procs, system monitor, user as directed (e.g., email, tweet)
  - App responds with desired action
    - Terminate/restart job, wait for checkpoint, etc.
- Checkpoint support
  - Timed or on-demand, binary or SCR
  - BB allocations, bleed/storage directives across hierarchical storage
- Restart support
  - From remote NVM checkpoint, relocate checkpoint, etc.

PMIx: Status

- Version 1.0 released
  - Documentation, wiki still being updated
- Server integrations underway
  - SLURM – July?
  - ORCM/OMPI underway
  - PGAS/GASNet – will undertake late summer
- Version 2.0 API definition
  - Request for input circulated

Alternative Models: Supporting Hadoop & Cloud

- HPC
  - Optimize versus performance, single-tenancy
- Hadoop support
  - Optimize versus data locality
  - Multi-tenancy
  - Pre-location of data to NVM, pre-staging from data store
  - Dynamic allocation requests
- Cloud = capacity computing
  - Optimize versus cost
  - Multi-tenancy
  - Provisioning and security support for VMs

Workload Manager: Job Description Language

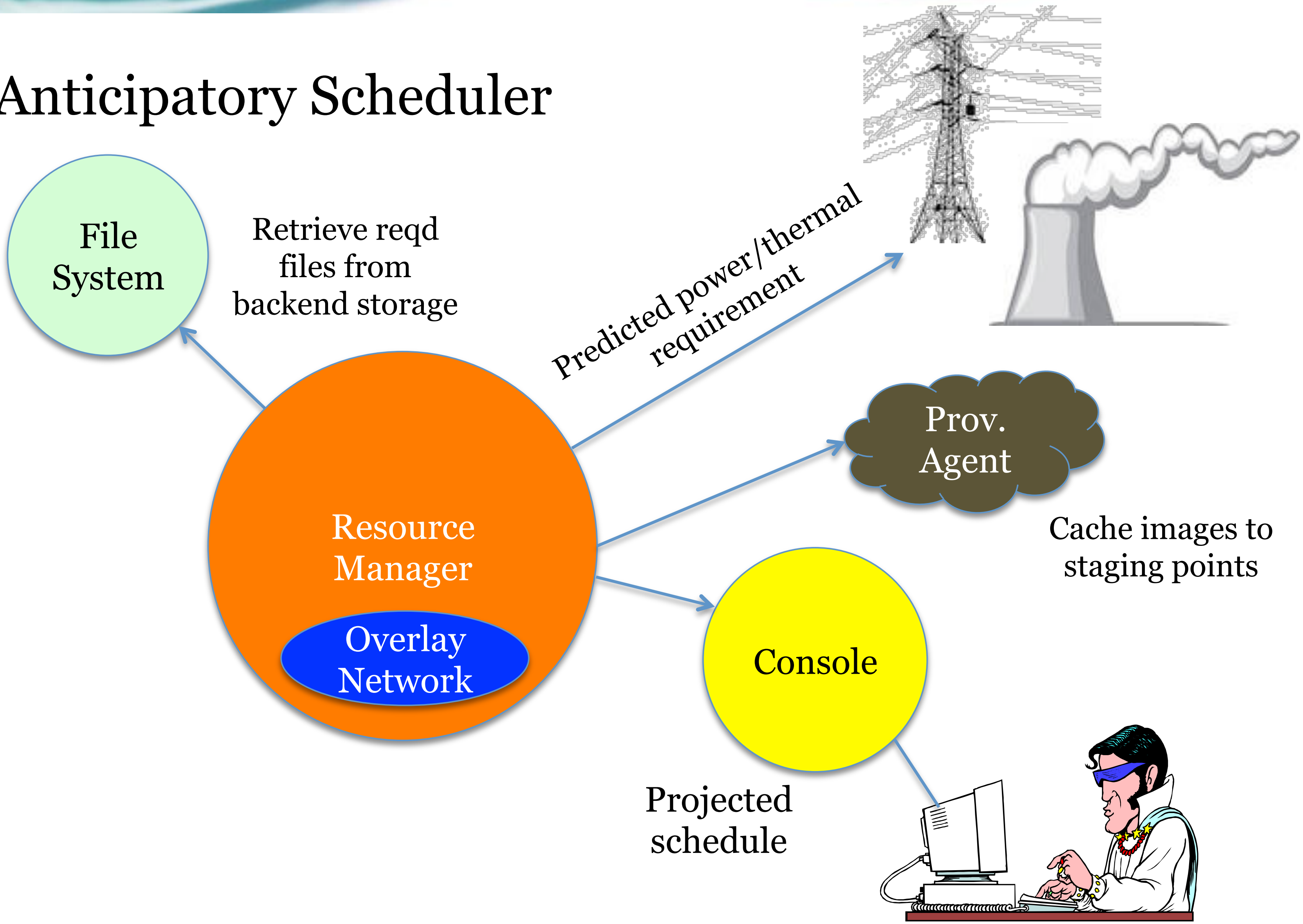
- Complexity of describing job is growing
  - Power, file/lib positioning
  - Performance vs capacity, programming model
  - System, project, application-level defaults
- Provide templates?
  - System defaults, with modifiers
    - --hadoop:mapper=foo,reducer=bar
  - User-defined
    - Application templates
    - Shared, group templates
  - Markup language definition of behaviors, priorities

30 yrs

Anticipatory Scheduling: Key to Success

- Current schedulers are reactionary
  - Compute next allocation when prior one completes
  - Mandates fast algorithm to reduce dead time
  - Limit what can be done in terms of pre-loading etc. as they add to dead time
- Anticipatory scheduler
  - Look ahead and predict range of potential schedules
  - Instantly select “best” when prior one completes
  - Support data and binary prepositioning in anticipation of most likely option
  - Improved optimization of resources as pressure on algorithm computational time is relieved

Anticipatory Scheduler



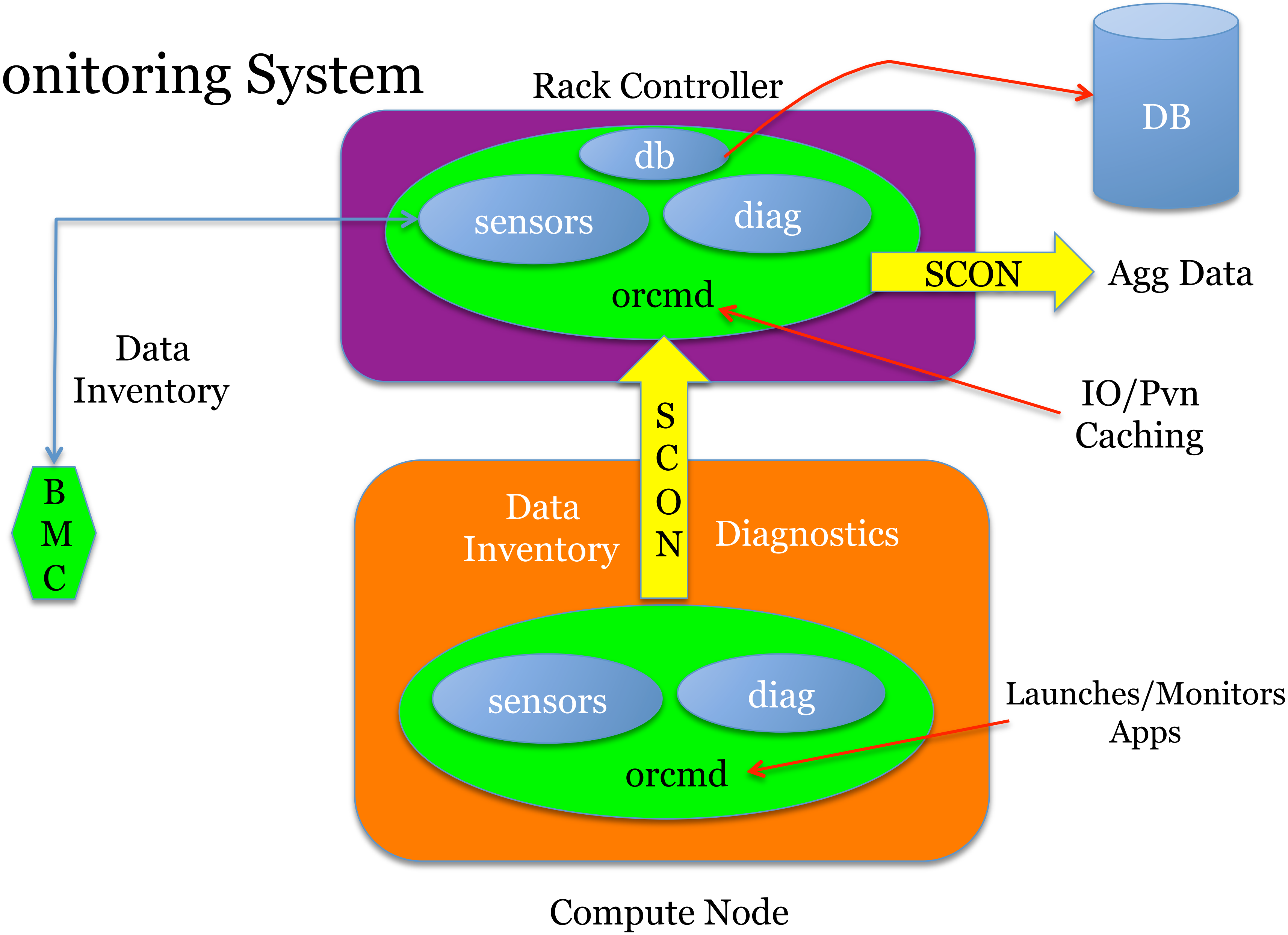
Controls Overview: A Definition

- Resource Manager
  - Workload manager
  - Run-time environment
- Monitoring
- Resiliency/Error Management
- Overlay Network
- Pub-sub Network
- Console

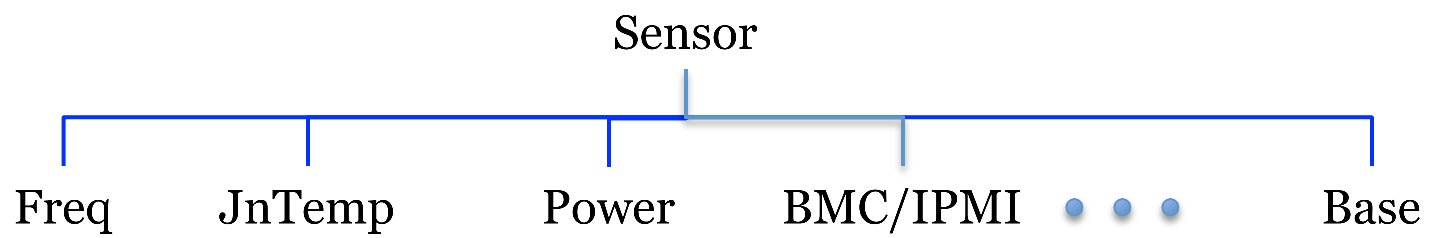
Key Functional Requirements

- Support all available data collection sensors
  - Environmental (junction temps, power)
  - Process usage statistics (cpu, memory, disk, network)
  - MCA, BMC events
- Support variety of backend databases
- Admin configuration
  - Select which sensors, how often sampled, how often reported, when and where data is stored
  - Severity/priority/definition of events
  - Intel provides default config
  - Local admin customizes config, on-the-fly changes

Monitoring System



Sensor Framework

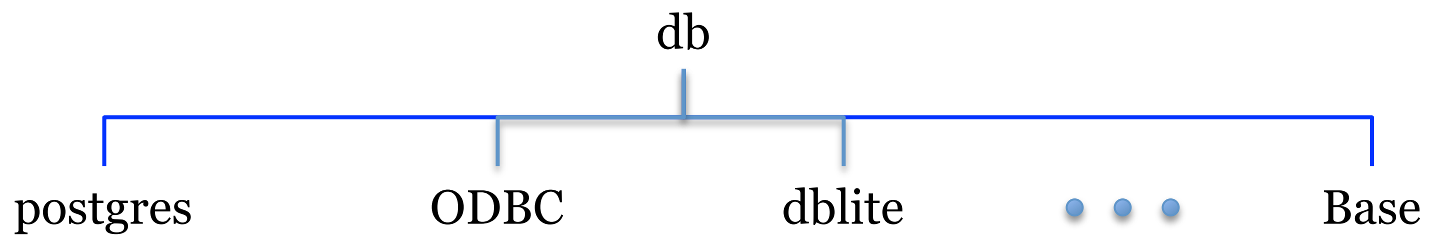


- Each sensor can operate in its own time base
  - Sample at independent rate
  - Output collected in common framework-level bucket
  - Can trigger immediate send of bucket for critical events
- Separate reporting time base
  - Send bucket to aggregator node at scheduled intervals
  - Aggregator receives bucket in sensor/base, extracts contributions from each sensor, passes to that sensor module for unpacking/recording

Inventory Collection/Tracking

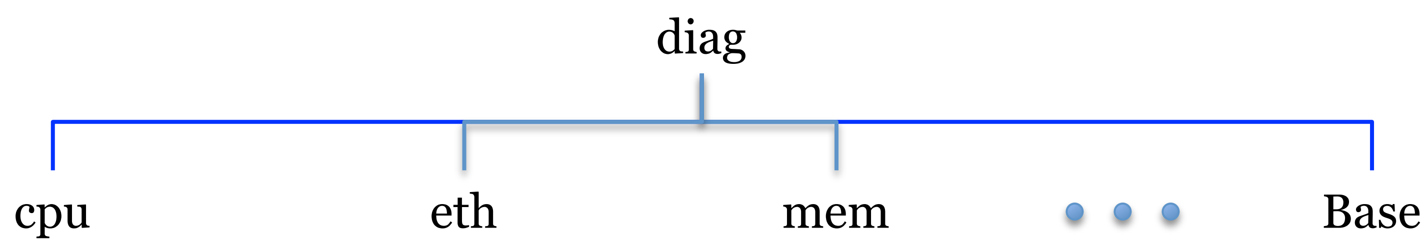
- Performed at boot
  - Update on-demand, warm change detected
- Data obtained from three sources
  - HWLOC topology collection (processors, memory, disks, NICs)
  - Sensor components (BMC/IPMI)
  - Network fabric managers (switches/routers)
- Store current inventory and updates
  - Track prior locations of each FRU
  - Correlate inventory to RAS events
  - Detect inadvertent return-to-service of failed units

Database Framework



- Database commands go to base "stub" functions
  - Cycle across active modules until one acknowledges handling request
  - API provides "attributes" to specify database, table, etc.
    - Modules check attributes to determine ability to handle
- Multi-threading supported
  - Each command can be processed by separate thread
  - Each database module can process requests using multiple threads

Diagnostics: Another Framework

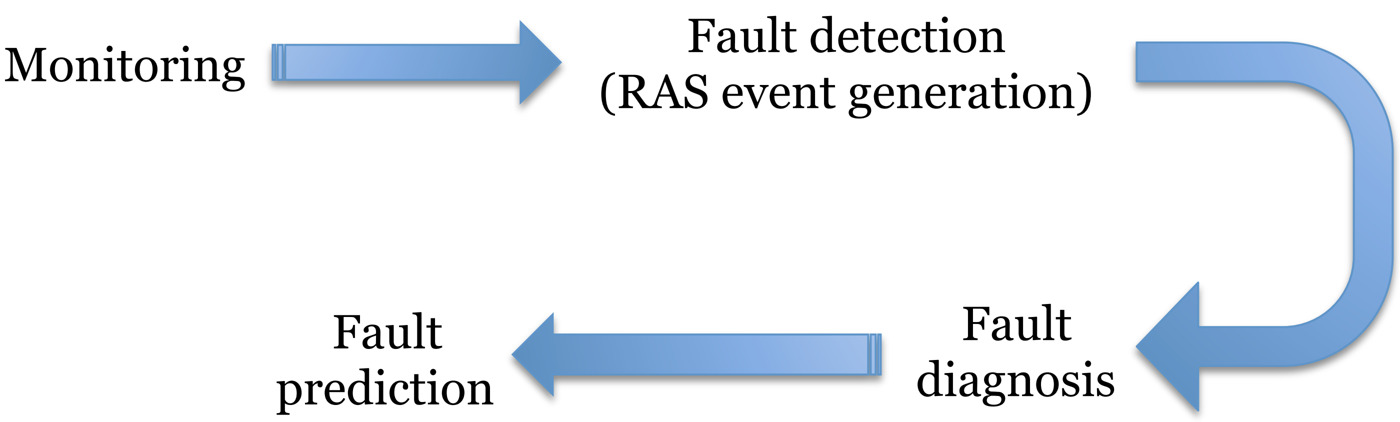


- Test the system on demand
  - Checks for running jobs and returns error
  - Execute at boot, when hardware changed, if errors reported, …
- Each plugin tests specific area
  - Data reported to database
  - Tied to FRUs as well as overall node

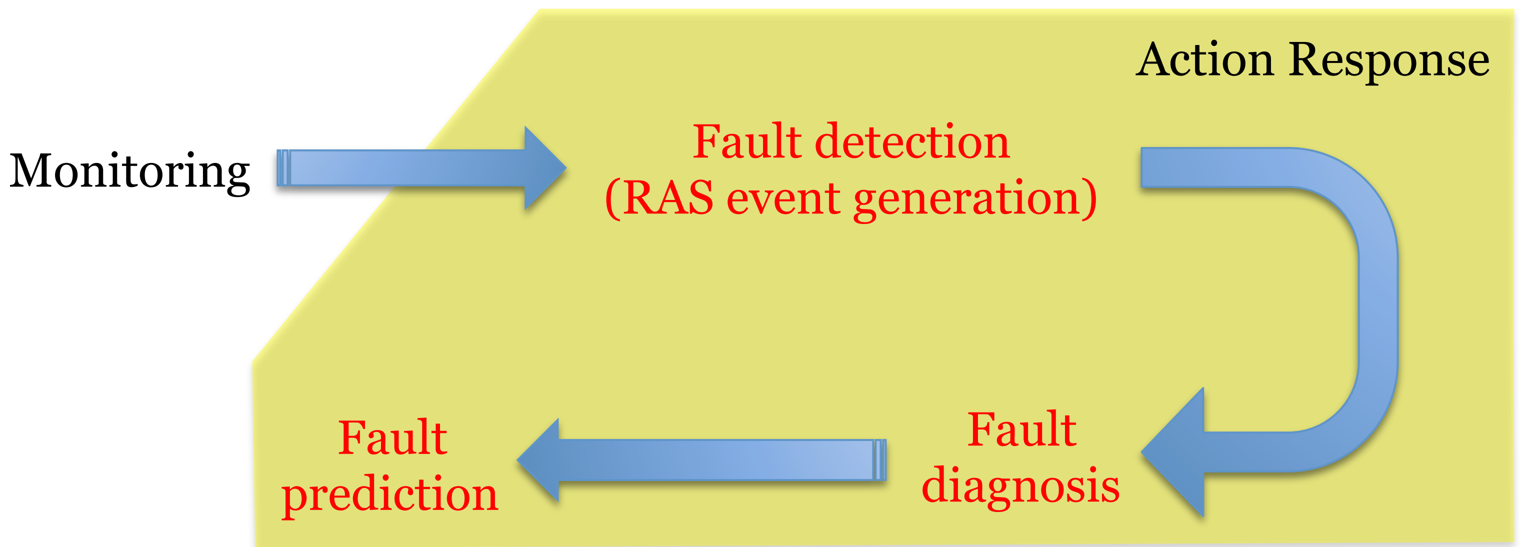
Controls Overview: A Definition

- Resource Manager
  - Workload manager
  - Run-time environment
- Monitoring
- Resiliency/Error Management
- Overlay Network
- Pub-sub Network
- Console

Planned Progression

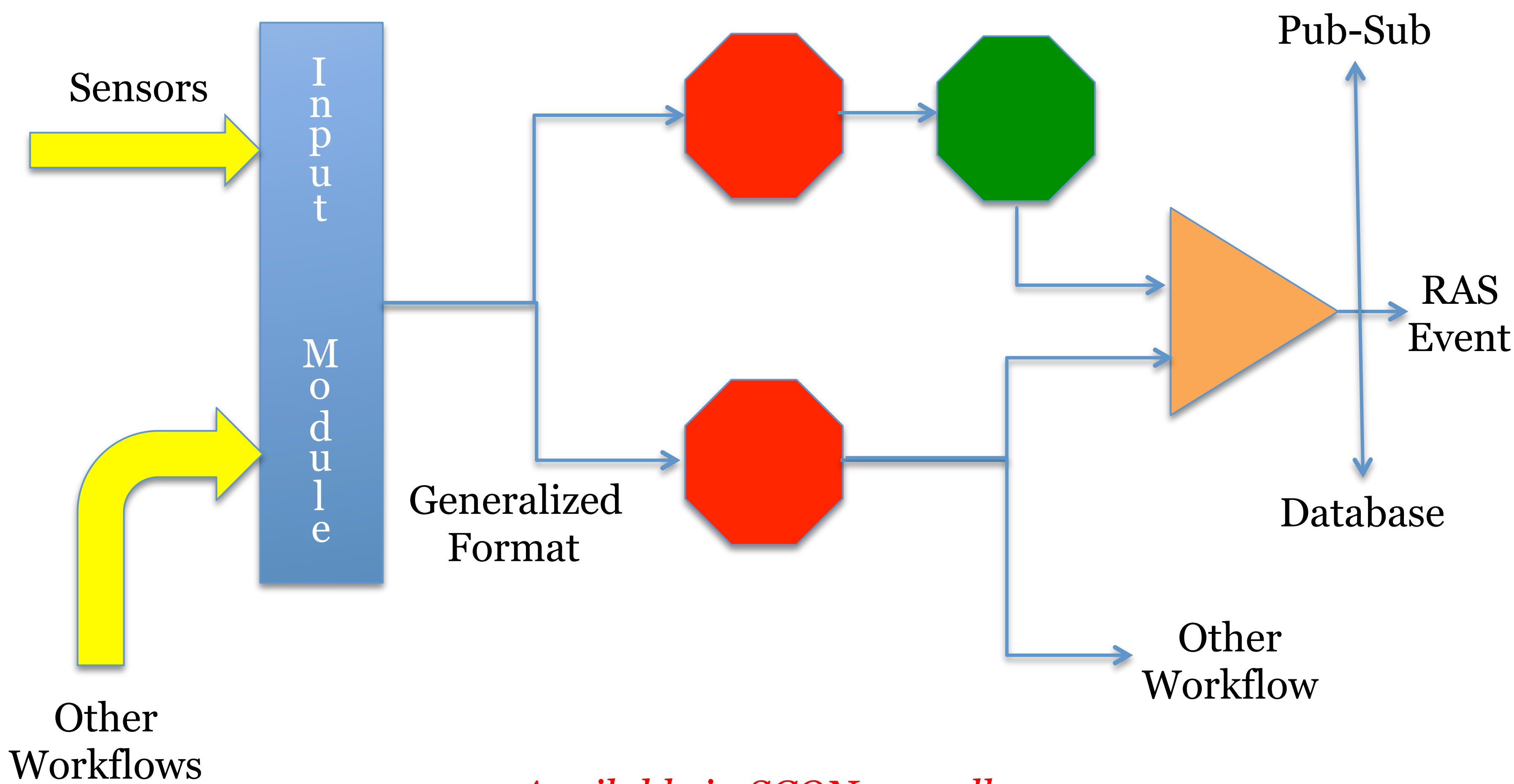


Planned Progression



FDDP

Analytics Workflow Concept



Available in SCON as well

Workflow Elements

- Average (window, running, etc.)
- Rate (convert incoming data to events/sec)
- Threshold (high, low)
- Filter
  - Selects input values based on provided params
- RAS event
  - Generates a RAS event corresponding to input description
- Publish data

Analytics

- Execute on aggregator nodes for in-flight reduction
  - Sys admin defines, user can define (if permitted)
- Event-based state machine
  - Each workflow in own thread, own instance of each plugin
  - Branch and merge of workflow
  - Tap stream between workflow steps
  - Tap data streams (sensors, others)
- Event generation
  - Generate events/alarms
  - Specify data to be included (window)

Distributed Architecture

- Hierarchical, distributed approach for unlimited scalability
  - Utilize daemons on rack/row controllers
- Analysis done at each level of the hierarchy
  - Support rapid response to critical events
  - Distribute processing load
  - Minimize data movement
- RM's error manager framework controls response
  - Based on specified policies

Fault Diagnosis

- Identify root cause and location
  - Sometimes obvious – e.g., when direct measurement
  - Other times non-obvious
    - Multiple cascading impacts
    - Cause identified by multi-sensor correlations (indirect measurement)
    - Direct measurement yields early report of non-root cause
    - Example: power supply fails due to borderline cooling + high load
- Estimate severity
  - Safety issue, long-term damage, imminent failure
- Requires in-depth understanding of hardware

Fault Prediction: Methodology

- Exploit access to internals
  - Investigate optimal location, number of sensors
  - Embed intelligence, communications capability
- Integrate data from all available sources
  - Engineering design tests
  - Reliability life tests
  - Production qualification tests
- Utilize learning algorithms to improve performance
  - Both embedded, post process
  - Seed with expert knowledge

Fault Prediction: Outcomes

- Continuous update of mean-time-to-preventative-maintenance
  - Feed into projected downtime planning
  - Incorporate into scheduling algo
- Alarm reports for imminent failures
  - Notify impacted sessions/applications
  - Plan/execute preemptive actions
- Store predictions
  - Algorithm improvement

Error Manager

- Log errors for reporting, future analysis
- Primary responsibility: fault response
    - Contains defined response for given types of faults
    - Responds to faults by shifting resources, processes
- Secondary responsibility: resilience strategy
    - Continuously update and define possible response options
    - Fault prediction triggers pre-emptive action
- Select various response strategies via component
    - Run-time, configuration, or on-the-fly command

Example: Network Communications

Network interface card repeatedly loses connection or shows data loss

Fault tolerant

- Modify run-time to avoid automatic abort upon loss of communication
- Detect failure of any on-going communication
- Reconnect quickly
- Resend lost messages from me
- Request resend of lost messages to me

Resilient

- Estimate probability of failure during session
  - Failure history
  - Monitor internals
    - Temperature, …
- Take preemptive action
  - Reroute messages via alternative transports
  - Coordinated move of processes to another node

Example: Node Failure

Node fails during execution of high-priority application

Fault tolerant

- Detect failure of process(es) on that node
- Find last checkpoint
- Restart processes on another node at the checkpoint state
  - May (likely) require restarting all processes at same state
- Resend any lost messages

Resilient

- Monitor state-of-health of node
  - Temperature of key components, other signatures
- Estimate probability of failure during future time interval(s)
- Take preemptive action
  - Direct checkpoint/save
  - Coordinate move with application
- Avoids potential need to reset ALL processes back to earlier state!

Controls Overview: A Definition

- Resource Manager
  - Workload manager
  - Run-time environment
- Monitoring
- Resiliency/Error Management
- Overlay Network
- Pub-sub Network
- Console

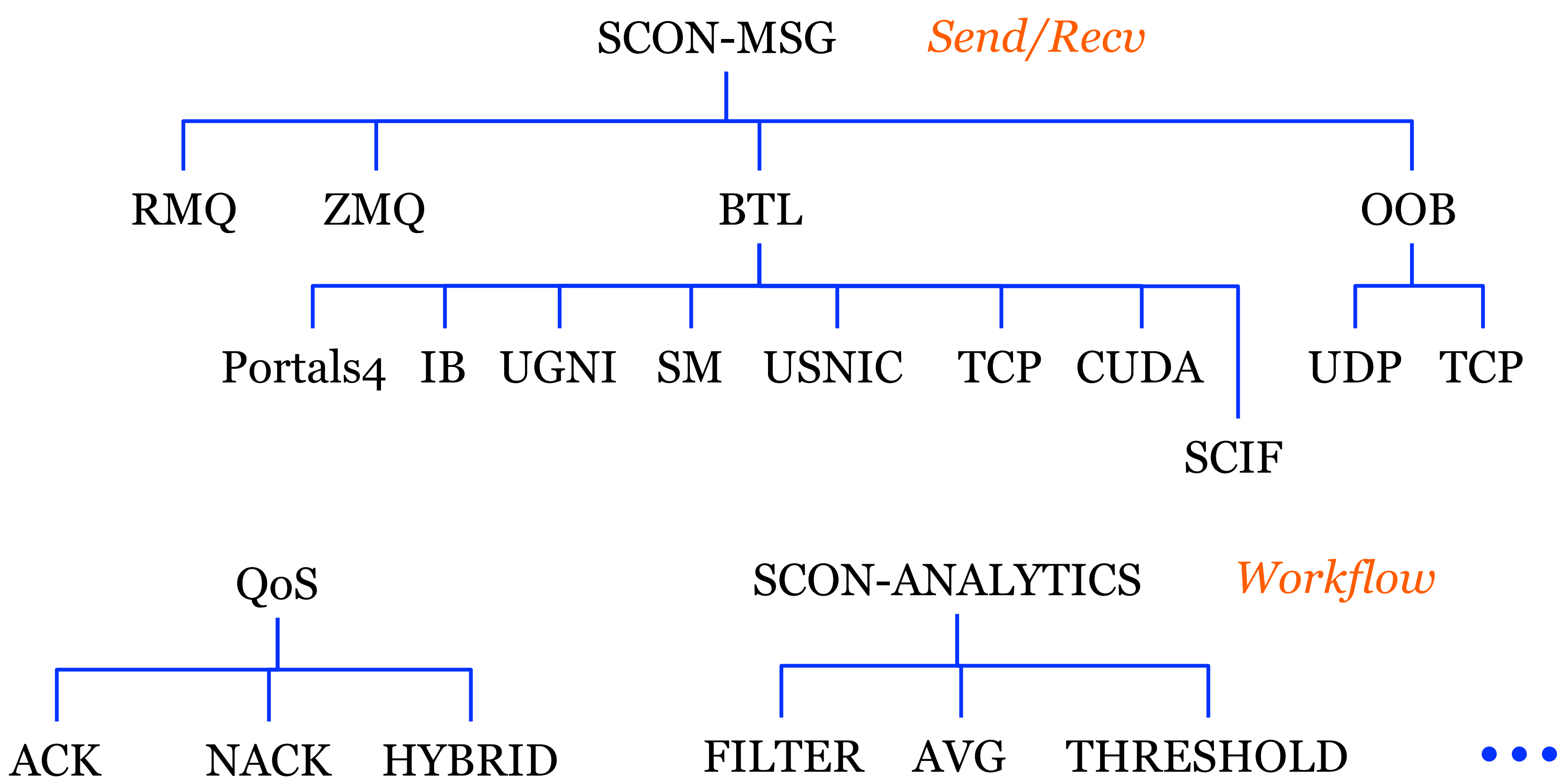
Definition: Overlay Network

- Messaging system
  - Scalable/resilient communications
  - Integration-friendly with user applications, system management software
  - Quality of service support
- In-flight analytics
  - Insert analytic workflows anywhere in the data stream
  - Tap data stream at any point
  - Generate RAS events from data stream

Requirements

- Scalable to exascale levels
  - Better-than-linear scaling of broadcast
- Resilient
  - Self-heal around failures
  - Reintegrate recovered resources
  - Support quality of service (QoS) levels
- Dynamically configurable
  - Sense and adapt, user-directable
  - On-the-fly updates
- In-flight analytics
  - User-defined workflows
  - Distributed, hierarchical analysis of sensor data to identify RAS events
  - Used by error manager
- Multi-fabric
  - OOB Ethernet
  - In-band fabric
  - Auto-switchover for resilience, QoS
- Open source (non-GPL)

High-Level Architecture



Messaging System

- Message management level
  - Manages assignment of messages (and/or fragments) to transport layers
  - Detect/redirects messages upon transport failure
  - Interfaces to transports
  - Matching logic
- Transport level
  - Byte-level message movement
  - May fragment across wires within transport
  - Per-message selection policies (system default, user specified, etc)

Quality of Service Controls

- Plugin architecture
  - Selected per transport, requested quality of service
- ACK-based (cmd/ctrl)
  - ACK each message, or window of messages, based on QoS
  - Resend or return error – QoS specified policy and number of retries before giving up
- NACK-based (streaming)
  - NACK if message sequence number is out of order indicating lost message(s)
  - Request resend or return error, based on QoS
  - May ACK after N messages
- Security level
  - Connection authorization, encryption, …

Controls Overview: A Definition

- Resource Manager
  - Workload manager
  - Run-time environment
- Monitoring
- Resiliency/Error Management
- Overlay Network
- Pub-sub Network
- Console

Pub-Sub: Two Models

- Client-Server (MQ)
  - Everyone publishes data to one or more servers
  - Clients subscribe to server
  - Server pushes data matching subscription to clients
  - Option: server logs and provides history prior to subscription
- Broker (MRNet)
  - Publishers register data with server
  - Clients log request with server
  - Server connects client to publisher
  - Publisher directly provides data to clients
  - Up to publisher to log, provide history

Pub-Sub: Two Models

- Client-Server (MQ)
  - Everyone publishes data to one or more servers
  - Clients subscribe to server
  - Server pushes data matching sub[illegible]ents
  - Option: server logs and [illegible]y prior to subscription
- Broker (MRNet)
  - Publishe[illegible]ata with server
  - Clients log [illegible]equest with server
  - Server connects client to publisher
  - Publisher directly provides data to clients
  - Up to publisher to log, provide history

Must limit #pubs, #subs

Pub-Sub: Two Models

ORCM: Support Both Models

- Client-Server (MQ)
  - Everyone publishes data to one or more servers
  - Clients subscribe to server
  - Server pushes data matching subscription to clients
  - Option: server logs and provides history prior to subscription
- Broker (MRNet)
  - Publishers register data with server
  - Clients log request with server
  - Server connects client to publisher
  - Publisher directly provides data to clients
  - Up to publisher to log, provide history

ORCM Pub-Sub Architecture

- Abstract, extendable APIs
  - Utilize “attributes” to specify desired data and other parameters
  - Allows each component to determine ability to support request
- Internal lightweight version
  - Support for smaller systems
  - When “good enough” is enough
  - Broker and client-server architectures
- External heavyweight versions supported via plugins
  - RabbitMQ
  - MRNet

ORCM Pub/Sub API

- Required functions
  - Advertise available event/data (publisher)
  - Publish event/data (publish)
  - Get catalog of published events (subscriber)
  - Subscribe for event(s)/data (subscriber asynchronous callback based and polling based)
  - Unsubscribe for event/data (subscriber)
- Security – Access control for requested event/data
- Possible future functions
  - QoS
  - Statistics

Controls Overview: A Definition

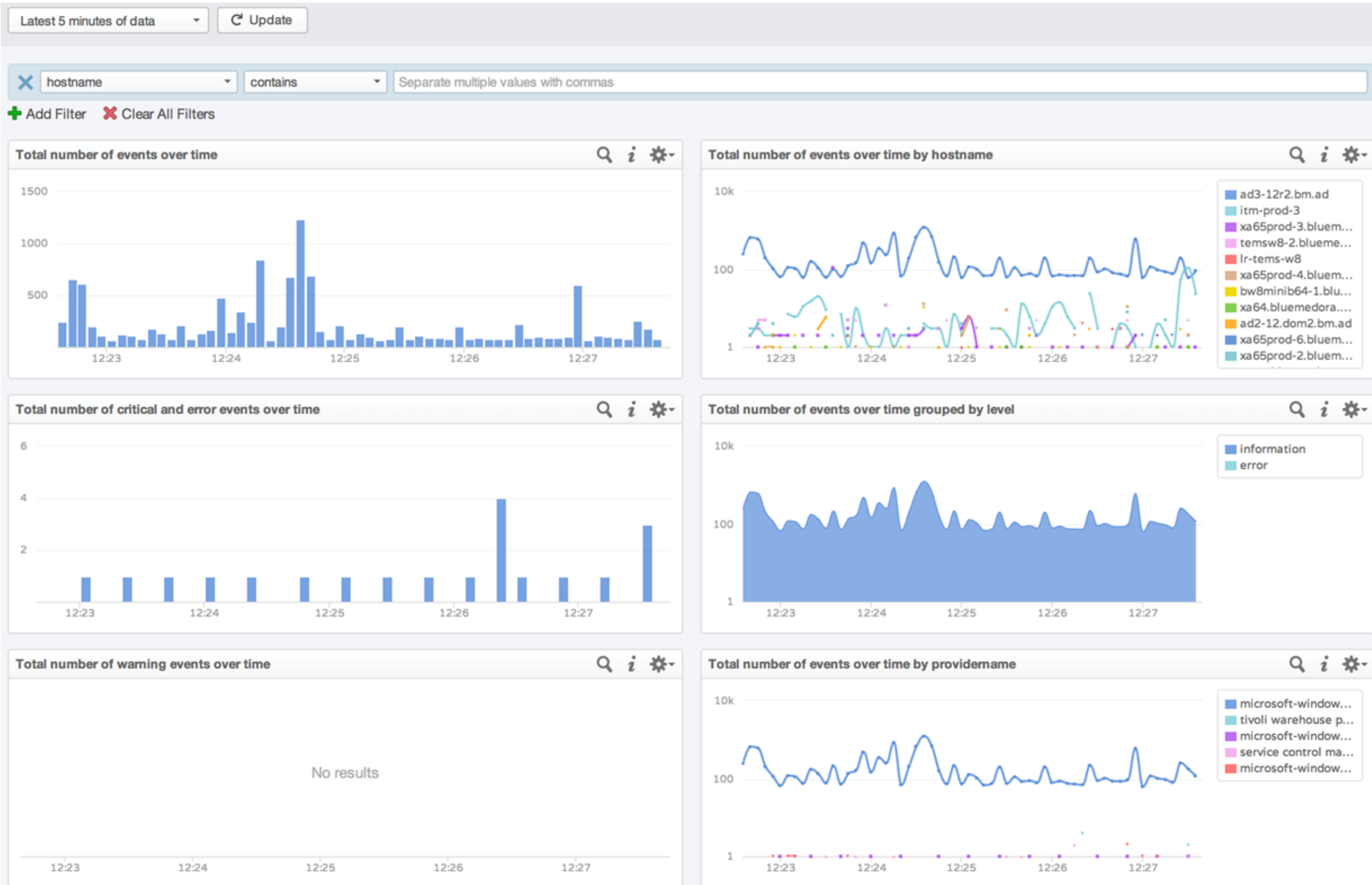
- Resource Manager
  - Workload manager
  - Run-time environment
- Monitoring
- Resiliency/Error Management
- Overlay Network
- Pub-sub Network
- Console

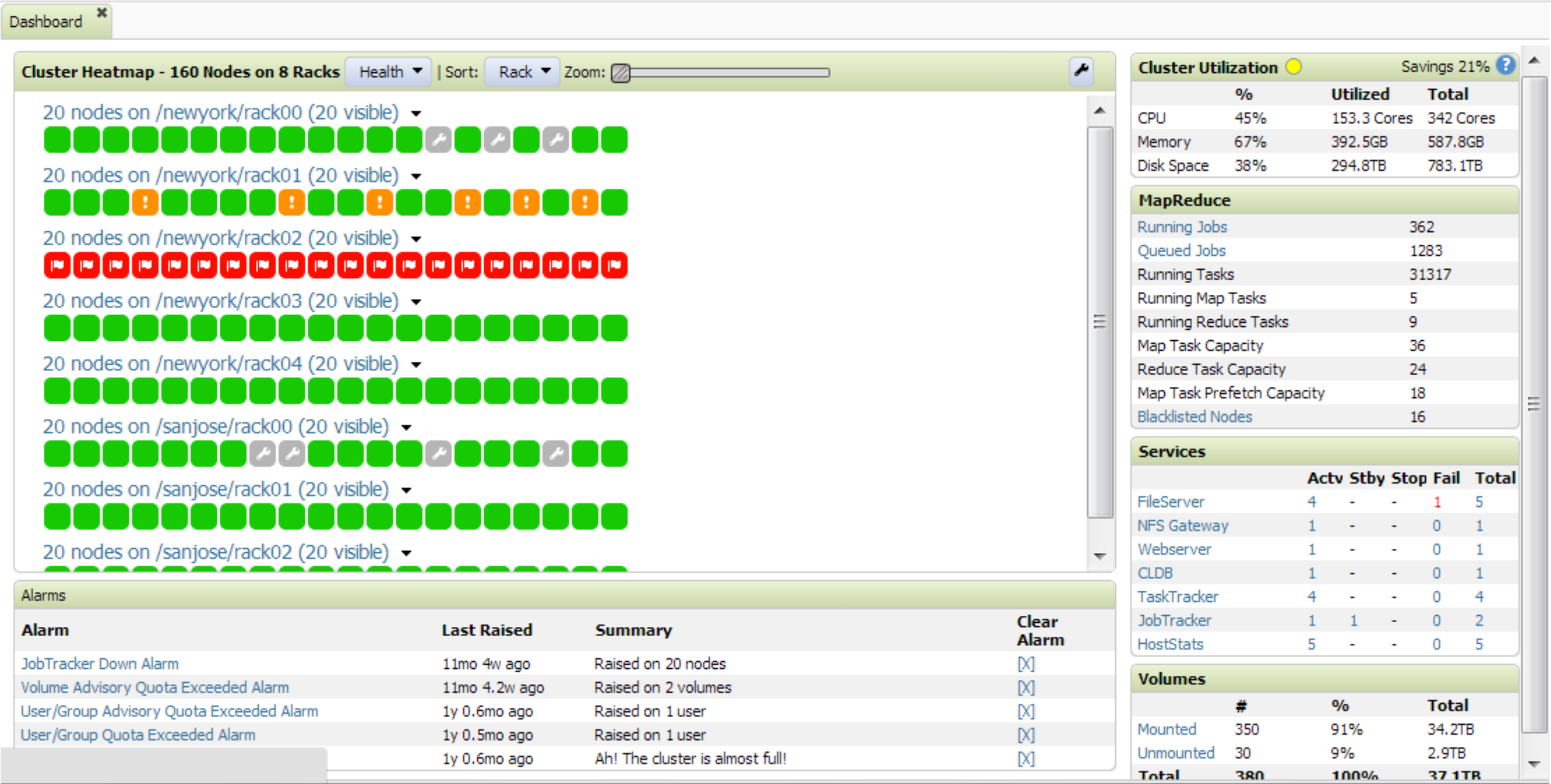
Basic Approach

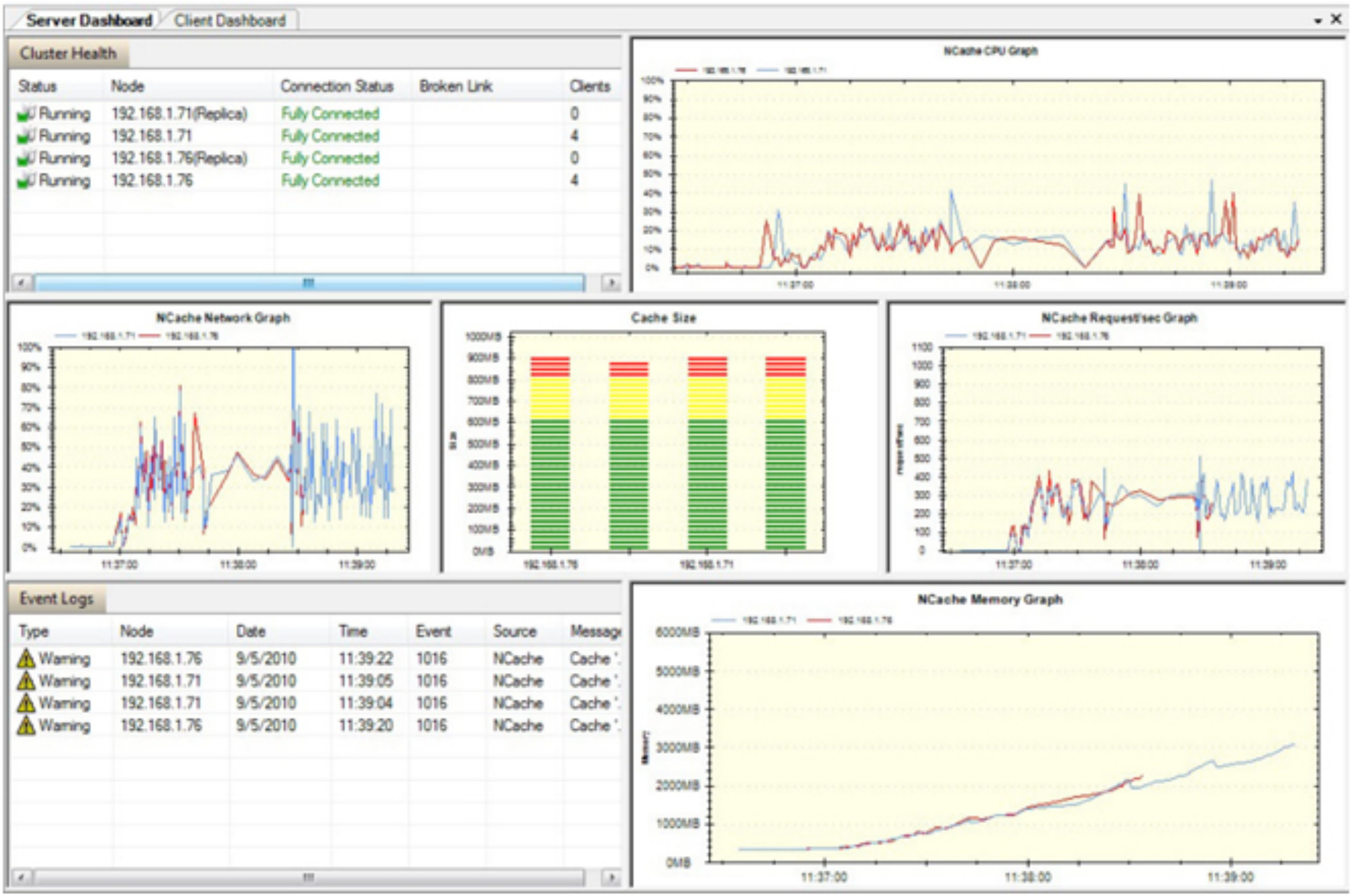
- Command-line interface
  - Required for mid- and high-end systems
  - Control on-the-fly adjustments, update/customize config
  - Custom code
- GUI
  - Required for some market segments
  - Good for visualizing cluster state, data trends
  - Provide centralized configuration management
  - Integrate with existing open source solution

Configuration Management

- Central interface for all configuration
  - Eliminate frustrating game of "whack-a-mole"
  - Ensure consistent definition across system elements
  - Output files/interface to controllers for individual software packages
    - RM (queues), network, file system
- Database backend
  - Provide historical tracking
  - Multiple methods for access, editing
- Command line and GUI
  - Dedicated config management tools


Latest 5 minutes of data
Update
hostname
contains
Separate multiple values with commas
Add Filter
Clear All Filters
Total number of events over time
1500
1000
500
12:23
12:24
12:25
12:26
12:27
Total number of events over time by hostname
10k
100
1
ad3-12r2.bm.ad
itm-prod-3
xa65prod-3.bluem...
temsw8-2.blueme...
lr-tems-w8
xa65prod-4.bluem...
bw8minib64-1.blu...
xa64.bluemedora....
ad2-12.dom2.bm.ad
xa65prod-6.bluem...
xa65prod-2.bluem...
Total number of critical and error events over time
6
4
2
Total number of events over time grouped by level
information
error
Total number of warning events over time
No results
Total number of events over time by providername
microsoft-window...
tivoli warehouse p...
microsoft-window...
service control ma...
microsoft-window...


Dashboard
Cluster Heatmap - 160 Nodes on 8 Racks
Health
Sort:
Rack
Zoom:
20 nodes on /newyork/rack00 (20 visible)
20 nodes on /newyork/rack01 (20 visible)
20 nodes on /newyork/rack02 (20 visible)
20 nodes on /newyork/rack03 (20 visible)
20 nodes on /newyork/rack04 (20 visible)
20 nodes on /sanjose/rack00 (20 visible)
20 nodes on /sanjose/rack01 (20 visible)
20 nodes on /sanjose/rack02 (20 visible)
Alarms
Alarm
Last Raised
Summary
Clear Alarm
JobTracker Down Alarm
11mo 4w ago
Raised on 20 nodes
[X]
Volume Advisory Quota Exceeded Alarm
11mo 4.2w ago
Raised on 2 volumes
[X]
User/Group Advisory Quota Exceeded Alarm
1y 0.6mo ago
Raised on 1 user
[X]
User/Group Quota Exceeded Alarm
1y 0.5mo ago
Raised on 1 user
[X]
1y 0.6mo ago
Ah! The cluster is almost full!
[X]
Cluster Utilization
Savings 21%
%
Utilized
Total
CPU
45%
153.3 Cores
342 Cores
Memory
67%
392.5GB
587.8GB
Disk Space
38%
294.8TB
783.1TB
MapReduce
Running Jobs
362
Queued Jobs
1283
Running Tasks
31317
Running Map Tasks
5
Running Reduce Tasks
9
Map Task Capacity
36
Reduce Task Capacity
24
Map Task Prefetch Capacity
18
Blacklisted Nodes
16
Services
Actv
Stby
Stop
Fail
Total
FileServer
4
-
-
1
5
NFS Gateway
1
-
-
0
1
Webserver
1
-
-
0
1
CLDB
1
-
-
0
1
TaskTracker
4
-
-
0
4
JobTracker
1
1
-
0
2
HostStats
5
-
-
0
5
Volumes
#
%
Total
Mounted
350
91%
34.2TB
Unmounted
30
9%
2.9TB



Server Dashboard
Client Dashboard
Cluster Health
Status
Node
Connection Status
Broken Link
Clients
Running
192.168.1.71(Replica)
Fully Connected
0
Running
192.168.1.71
Fully Connected
4
Running
192.168.1.76(Replica)
Fully Connected
0
Running
192.168.1.76
Fully Connected
4
NCache CPU Graph
NCache Network Graph
Cache Size
NCache Request/sec Graph
Event Logs
Type
Node
Date
Time
Event
Source
Messag
Warning
192.168.1.76
9/5/2010
11:39:22
1016
NCache
Warning
192.168.1.71
9/5/2010
11:39:05
1016
NCache
Warning
192.168.1.71
9/5/2010
11:39:04
1016
NCache
Warning
192.168.1.76
9/5/2010
11:39:20
1016
NCache
NCache Memory Graph

HPC Controls