Adaptive distributed computing for fractionated space systems

Richard Golding, Theodore Wong
IBM Almaden Research Center

3 August 2006
What is adaptive distributed computing?

- Distributed application uses resources at multiple nodes
- App must adapt as needs and resources change
- Automatic, not manual
- Multiple applications sharing resources
Examples

- Data analysis for mission (re)planning
  - Use as many compute cycles as possible
  - Do as much as possible in planning window
  - Experimenters/observers adding new analyses

- Communication up/downlink and inter-node channels

- Data storage for later transmission or analysis
  - Store extra copies when extra space
  - Retain high-priority data when running short
- Application consists of multiple agents, with different roles
- Middleware provides distributed environment
- Each local node provides predictable resources and services—compute, storage, communication, sensing, energy
Architecture (detail)
What can we draw upon?

- Grid and peer-to-peer systems
  - Supercomputing clusters, PlanetLab, MPI
- Autonomic/self-managing systems
  - Management tools
- Mobile adaptivity
- Group communication systems
  - Ensemble, Horus
- Virtual machines
  - VMware, Hypervisor
- Embedded/realtime OSes
  - Commercial and experimental
What’s new here?

• *(Why not use an existing system?)*

• Integration of security, realtime, fault containment in middleware

• Local resource scheduling across multiple resources

• Self-managed configuration

• API for adaptive applications
Aspects of the problem

- Resource allocation
- Groups and membership
- Communication
- Failure tolerance
- Security
- Programming model and tools
- Standardization and interoperability
Resource allocation

- Decision algorithm: constrained multidimensional bin-packing
- Isolation between applications
Groups and membership

- How are the pieces of an app identified? What defines an app?

- Key properties
  - Consistency
  - Resilience
  - Containment
  - Security
  - Scalability

- Research focus: self-identification, decentralization, loose synchronization
Failure tolerance

- detect
- reconfig
- recover

Fault containment
Group mechanisms
Resource allocation

- predict
- proactive reconfig

Failure predictors
Consumable/wear monitoring
Resource allocation

System monitoring
Ground analysis
Manufacture and launch

- predict
- prepare replacement
- inform supply chain

Time in future:
- now
- soon (minutes to hours)
- long-term (weeks to months)
Programming model and tools

- Language and API
- Development tools

- Test framework
  - Continuous test, fast regression test
- Simulators
- Model checkers
Standardization and interoperation

• Expect satellites added over time
  • Multiple vendors, versions
• Interoperation ⇒ standards

• Defining standards
  • Venue? when ready? formal models?
• Interoperability tests
  • Formal checking
  • Testing
Putting this together

Notional mission: multiple experiments, onboard mission (re)planning

Inspired by:

Contacts and information

- Richard Golding
  rgolding@us.ibm.com
  http://soe.ucsc.edu/~golding

- Theodore Wong
  theowong@us.ibm.com
  http://www.tmwong.org

- Chris Hanson
  cjhanson@us.ibm.com

- In conjunction with SSRC, UC Santa Cruz, and PDL, Carnegie Mellon Univ.
Backup slides
K2 distributed storage system

- Vehicle for research—*not* a product
- No central administration; federate when global view needed
- Delegate function to as low a level as possible
- Provide support to higher-level application management
Resource pools: external view

- A virtual collection of storage
- One per user or application
- Each pool is independent
- Specified by:
  - Capacity, Performance, Reliability
  - Reserve and limit

- Initially: capacity = bytes; performance = IO/s; reliability = MTTDL
Implementing pools

- Virtual pool backed by physical allocation pools
- Pools contain objects for storing user data
- Decision algorithm: how much to put where
- Storage server enforces resource allocation
Resource allocation decisions: normal

- Normal case: online decision for one pool
  - Creating or modifying a pool’s requirements
  - Load balancing
- Use constrained multidimensional bin packing
- Constraints derived from reliability requirements
Resource allocation decisions: failure

- Multi-pool assignment required
- Backtracking search for feasible solution (better is possible)
Resource allocation decisions: failure

- Multi-pool assignment required
- Backtracking search for feasible solution (better is possible)
Making decisions

- Each resource pool is an independent group
- APs elect a manager; manager watches over pool
- Manager is disposable
- Manager runs decision algorithm
- All information in allocation pools
Local resource management

- Goal: isolation between pools
- Capacity: just accounting
- Performance: requires scheduler
- Tradeoff: performance vs. efficiency
- Provides reserve and limit, plus fair sharing
- Working to add cache, network