Scalability of the Microsoft Cluster Service

Werner Vogels,
Dan Dumitriu, Ashutosh Agrawal,
Teck Chia, Katherine Guo

Reliable Distributed Systems Group

Dept. of Computer Science
Cornell University

Agenda

- Research Goals
- Intro into MS Cluster Service
- Practical Scalability
- Evaluation of MSCS components
- Conclusions
- What’s Cookin’?

Disclaimer©

- The tests have taken MSCS far beyond the goals set in its design.

- Any limitations are due to to pushing the technology to extremes, and are not present in the commercial systems.

Agenda

- **Research Goals**
- Intro into MS Cluster Service
- Practical Scalability
- Evaluation of MSCS components
- Conclusions
- What’s Cookin’?

Research Goals

General: **Reliable Distributed Systems**

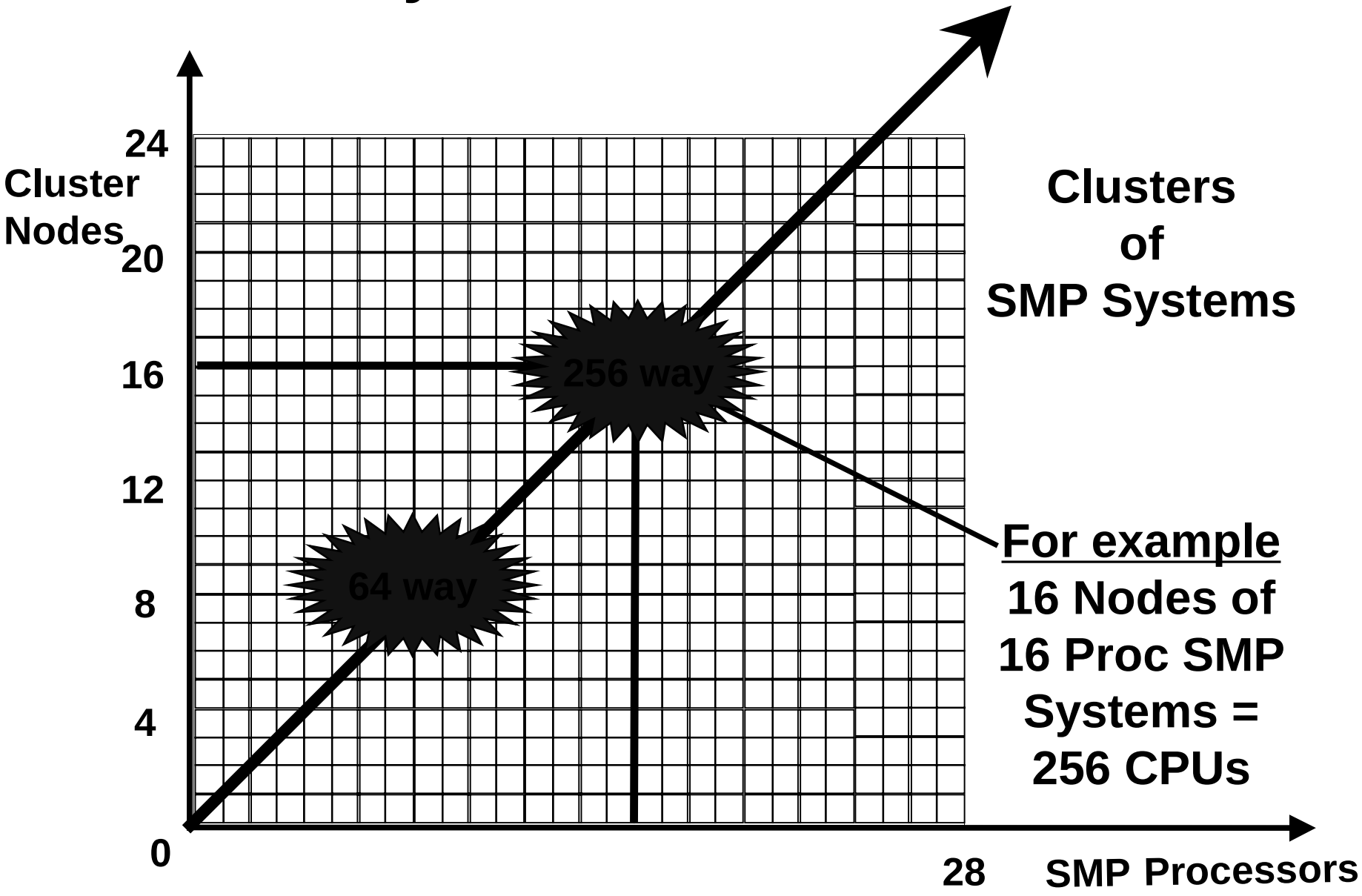
Specific Cluster Research:

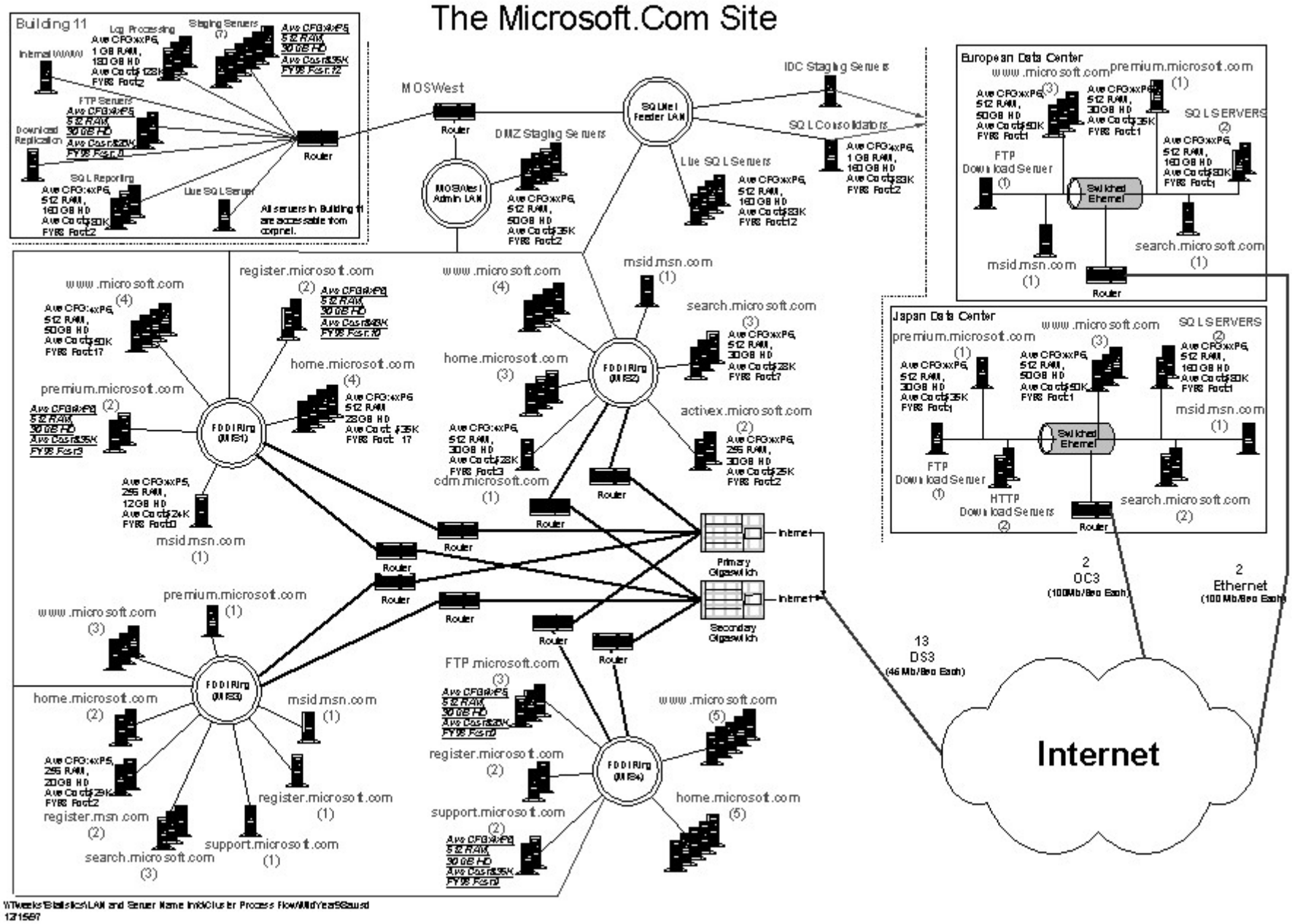
- Efficient Distributed Management
- Low Overhead Scalability
- Cluster Collections
- Cluster Aware Programming Tools (Quintet)

Research into Scalable Clusters

- Today's practice
  - Parallel Computing on 512++ nodes
  - High-Availability up to 16 nodes
- Distribution and Fault Management are very scale sensitive.
  - Failure Management
  - Node Membership
  - Cluster-Wide Consistency

The Reality of Scalable Clusters





\\Tweeks\Statistics\LAN and Server Name Info\Cluster Process Flow\MidYear98.vsd
12/15/97

Mandatory Reading

“In Search of Clusters

the ongoing battle in lowly parallel computing”

Gregory Pfister

second edition
Prentice Hall

Agenda

- Research Goals
- **Intro into MS Cluster Service**
- Practical Scalability
- Evaluation of MSCS components
- Conclusions
- What’s Cookin’?

Windows NT Clusters

What is clustering to Microsoft?

- Group of independent systems that appear as a single system
- Managed as a single system
- Common namespace
- Services are “cluster-wide”
- Ability to tolerate component failures
- Components can be added transparently to users
- Existing client connectivity is not effected by clustered applications

Windows NT Clusters
Development goals

- Extend Windows NT to seamlessly include cluster features
- Ship high-availability features for Windows NT first
  - *Support key applications without modification*
  - *Failover support for base Windows NT hardware, services, and applications*
  - *Available API for ISV products*
- Develop scalability product later

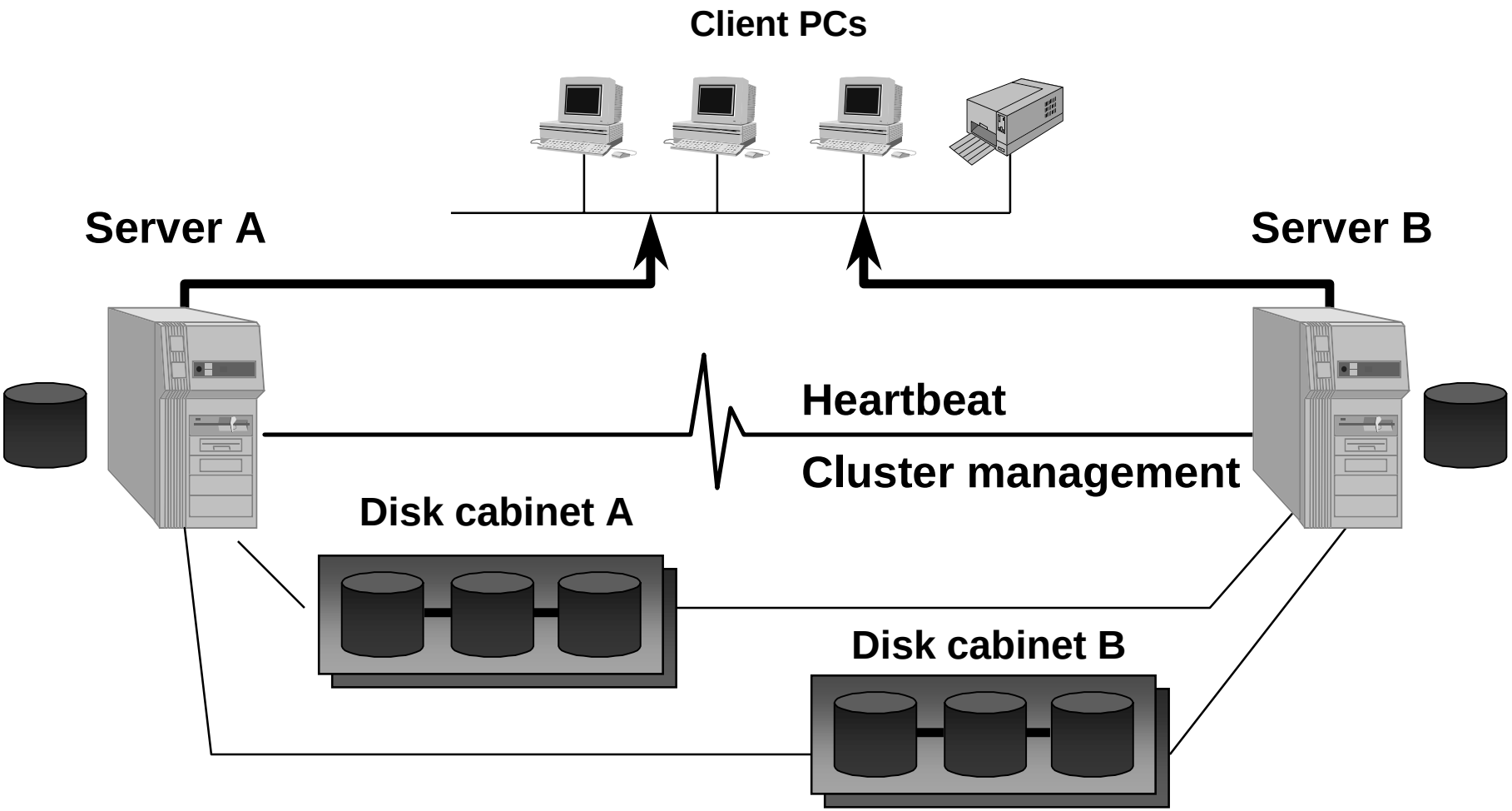
MSCS Features

- Shared nothing
  - Simplified hardware configuration
- Remoteable tools
- Windows NT manageability enhancements
  - Never take a “cluster” down: rolling upgrade
- Microsoft® BackOffice™ product support
- 3rd Party Support: SAP, Oracle

Non-Features Of MSCS

- Not lock-step/fault-tolerant
- Not able to “move” running applications
  - “MSCS” restarts applications that are failed over to other cluster members
- Not able to recover shared state between client and server (i.e., file position)
  - All client/server transactions should be atomic
  - Standard client/server development rules still apply

MSCS Cluster



Agenda

- Research Goals
- Intro into MS Cluster Service
- **Practical Scalability**
- Evaluation of MSCS components
- Conclusions
- What’s Cookin’?

Scaling Distributed Systems 101

- Reduce algorithmic dependency on the number of nodes.
- Traditional Solutions:
  - Reduce Synchronous Behavior
  - Reduce System Complexity
- Radical Solutions:
  - Epidemic (gossip, probabilistic) techniques

Scaling MSCS?

- Why do we care? *(Tools, Tools, Tools)*

- Do the Distributed Algorithms scale?
- Are there bottlenecks in the implementation?
- Is it a good basis for Cluster Aware Support?

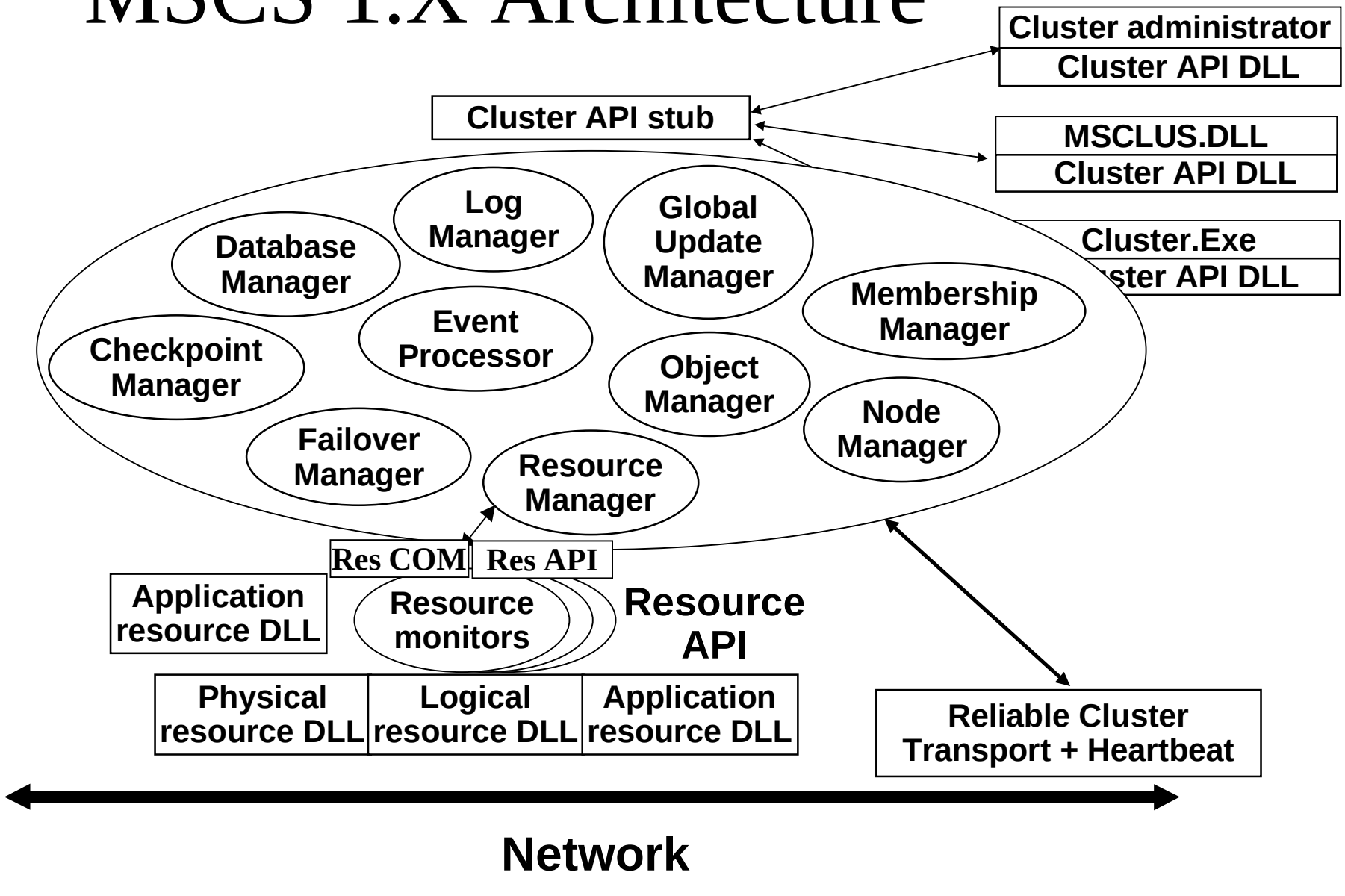
Agenda

- Research Goals
- Intro into MS Cluster Service
- Practical Scalability
- **Evaluation of MSCS components**
- Conclusions
- What’s Cookin’?

Cornell Test Cluster

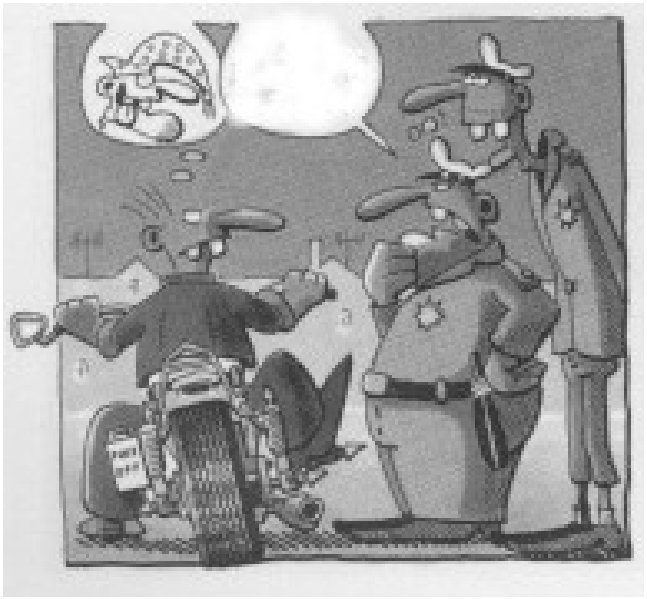
- 32 node MSCS Cluster
- Modified MSCS code
- 300 MHz PII - 200 P6 (128 Mb memory)
- 100 Mbit/sec Switched Ethernet
- Test environment
  - Unloaded systems
  - Loaded system with IO intensive Apps

MSCS 1.X Architecture



Components under Investigation

- Failure Detection
- Node Membership
  - Join operation
  - Reconfiguration after failure
- Consistent Distributed State Management

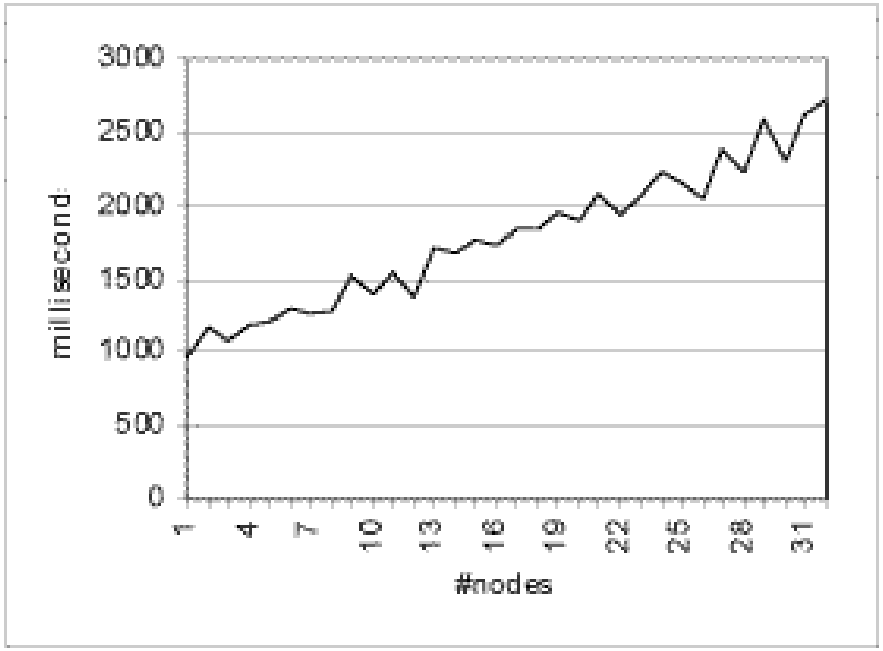


Failure Detection

- Heartbeat broadcast
  - over all interfaces
  - period 1.2 second
- Interface suspicion after 3 misses
- Node Suspicion after 6 misses (7.2 seconds)

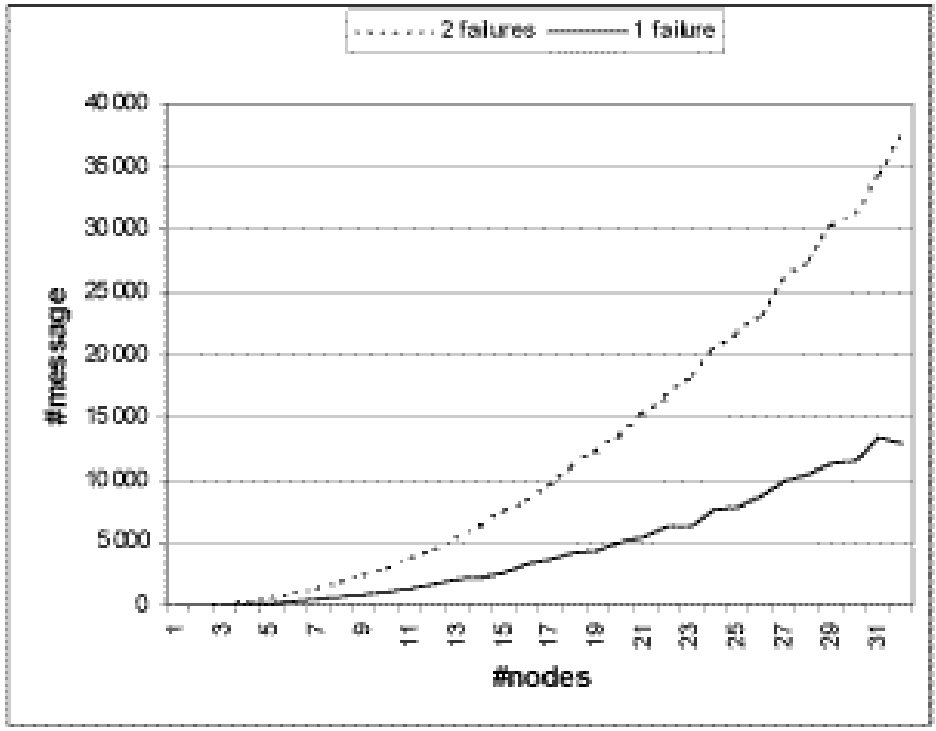
Membership Join

- 6 phase operation
  - discovery
  - lock
  - enable network
  - petition
  - database sync
  - unlock



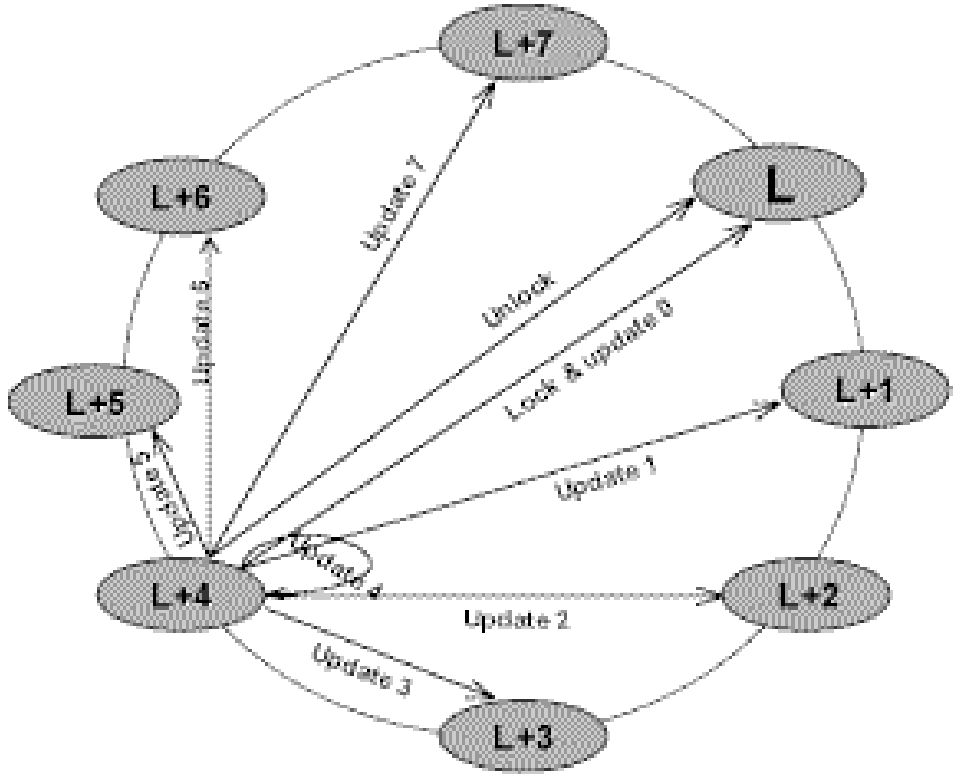
Membership Regroup

- 5 Phase fully distributed
  - Activate
  - Closing
  - Pruning
  - Cleanup phase one
  - Cleanup phase two



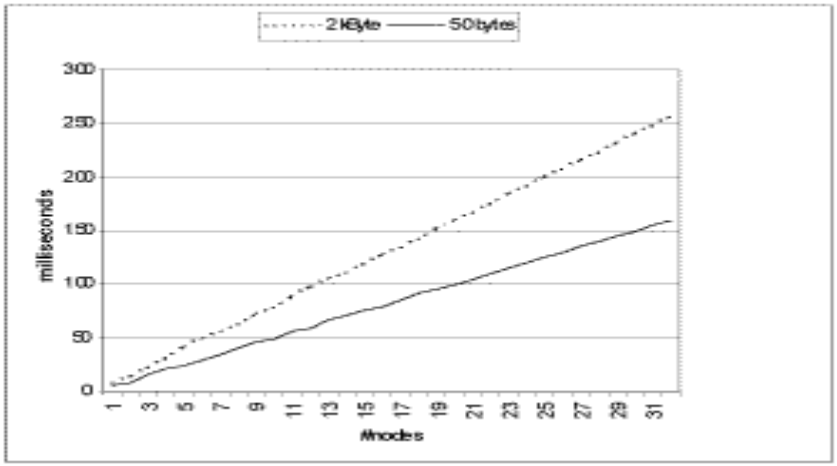
Global Update I

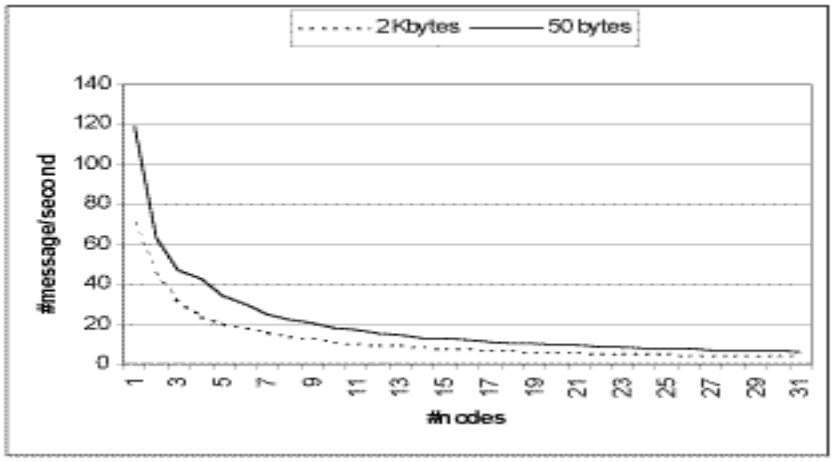
- Atomic / Total Order
  - Organize nodes in a ring
  - Acquire lock
  - Transmit to each node in order
  - Release lock
- Handles a number of failure scenarios



Global Update

- Developed for sparse updates of OS structures
- Implemented in MSCS using repeated RPC
- Collapses under load





Agenda

- Research Goals
- Intro into MS Cluster Service
- Practical Scalability
- Evaluation of MSCS components
- **Conclusions**
- What’s Cookin’?

Conclusions

- **Can the current Algorithms scale**?
  - *FD* & *Regroup*: Yes
  - *GUP*: 10-16 nodes
- **Are there bottlenecks in the implementation**?
  - *FD* & *Regroup*: Repeated p2p in
  - *Join* & *GUP*: RPC Trains
- **Is it a good basis for cluster aware support**
  - NO

Agenda

- Research Goals
- Intro into MS Cluster Service
- Practical Scalability
- Evaluation of MSCS components
- Conclusions
- **What’s Cookin’?**

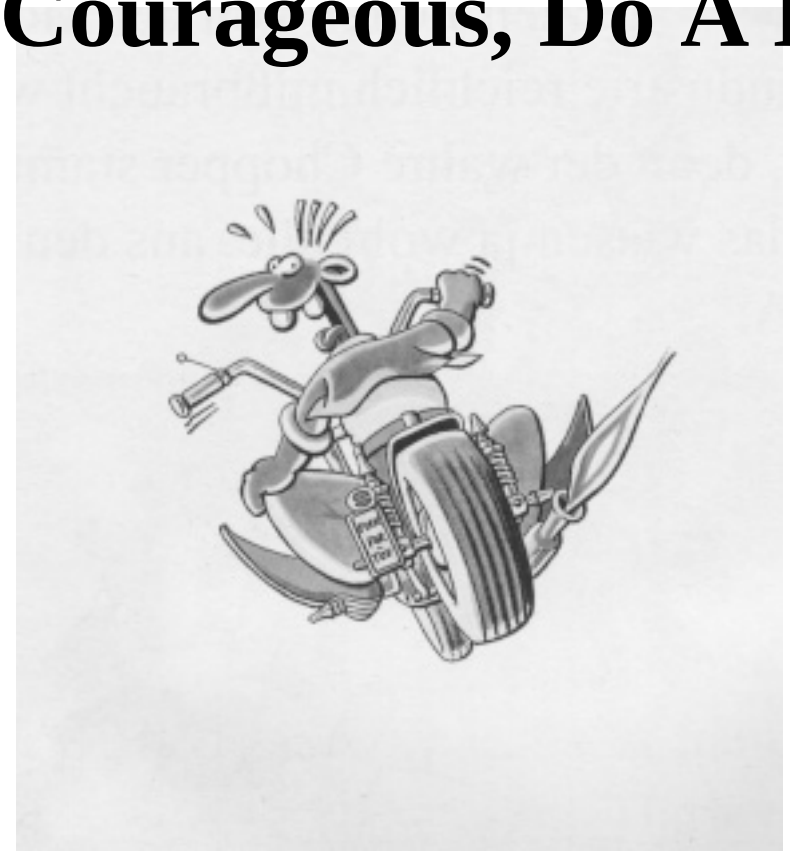
Rat Pack Clusters

A Quick Glance in the Kitchen

Rat Pack Clusters

- Tested on 200++ nodes
- Mixed Nuts: NT & Unix
- Provides Cluster Events
- Epidemic FD & Membership
- Probabilistic Communication Tools
- Sub-Clusters for Limited Scalability operations

Be Courageous, Do A Demo



Any Questions?