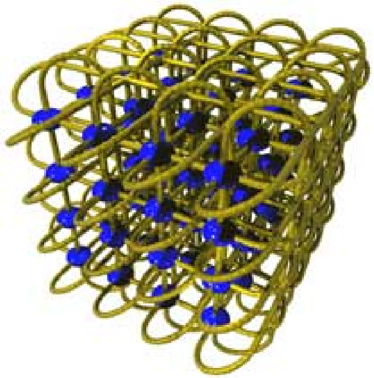
January, 2003



Super-scalable Algorithms

Next Generation Supercomputing on 100,000 and more Processors

Christian Engelmann

This document describes activities performed under contract number DE-AC0500OR22750 between the U.S. Department of Energy and Oak Ridge Associated Universities.

Overview

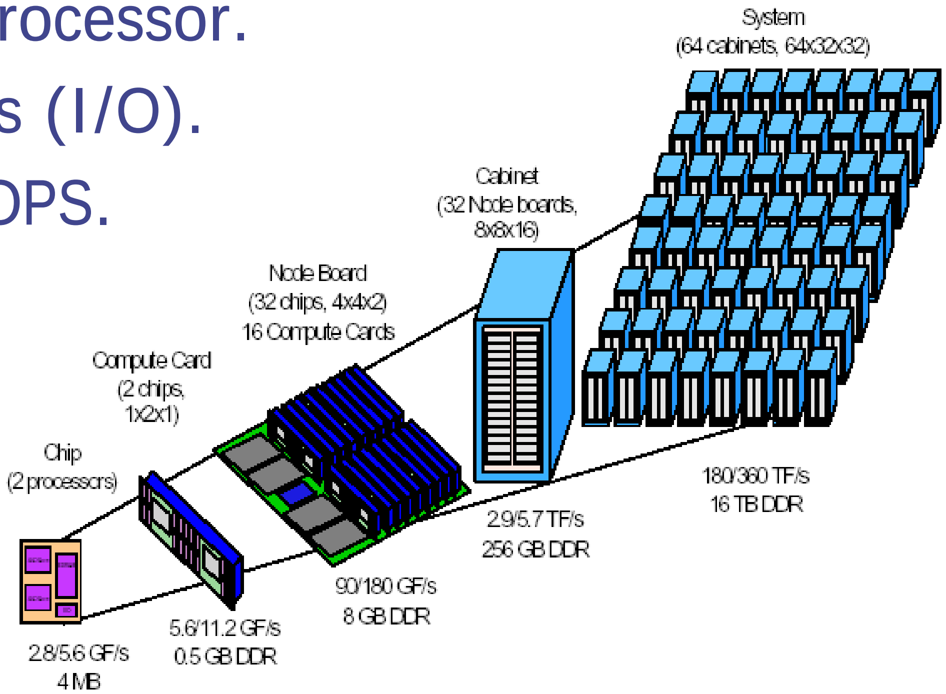
- Super-scale architectures.
- Scalability and fault-tolerance issues.
- Cellular algorithms theory.
- ORNL/IBM collaboration.
- IBM BlueGene\L emulators.
- ORNL cellular architecture simulator.
- Super-scalable algorithms.
- Super-scalable diskless checkpointing.
- Conclusions and ideas for the future.

Super-scale Architectures

- Current tera-scale supercomputers have up to 10,000 processors.
- Next generation peta-scale systems will have 100,000 processors and more.
- Such machines may easily scale up to 1,000,000 processors in the next decade.
- IBM currently builds the BlueGene\L at Lawrence Livermore National Laboratory.

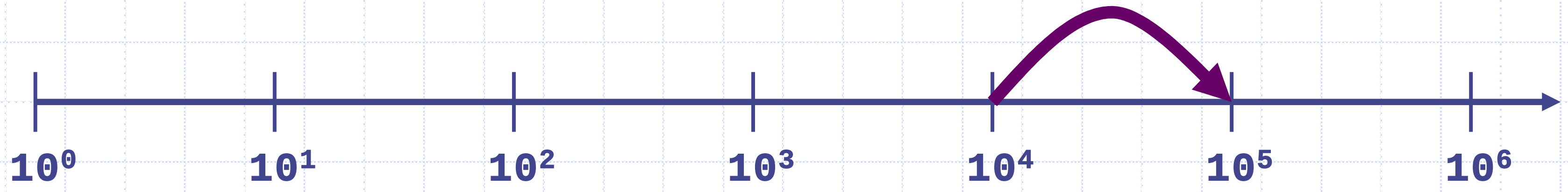
IBM BlueGene\L at LLNL

- Up to 64K diskless nodes with 2 processors per node.
- Only 256MB RAM per processor.
- Additional service nodes (I/O).
- Estimated 360 Tera FLOPS.
- Over 150k processors.
- Global tree network.
- 3-D torus network.
- Gigabit Ethernet.
- Operational in 2005.



Scalability Issues

- How to make use of 100,000 processors?
- System scale jumps by a magnitude.
- Current algorithms do not scale well on existing 10,000-processor systems.
- Next generation peta-scale systems are useless if efficiency drops by a magnitude.

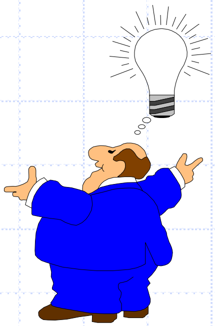


Fault-tolerance Issues

- How to survive on 100,000 processors?
- Failure rate grows with the system size.
- Mean time between failures may be a few hours or just a few minutes.
- Current solutions for fault-tolerance rely on checkpoint/restart mechanisms.
- Checkpointing 100,000 processors to central stable storage is not feasible anymore.

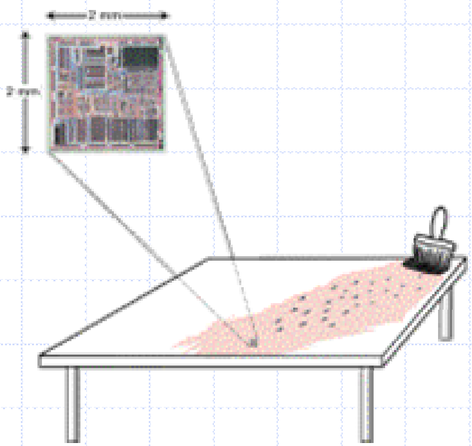
Cellular Algorithms Theory

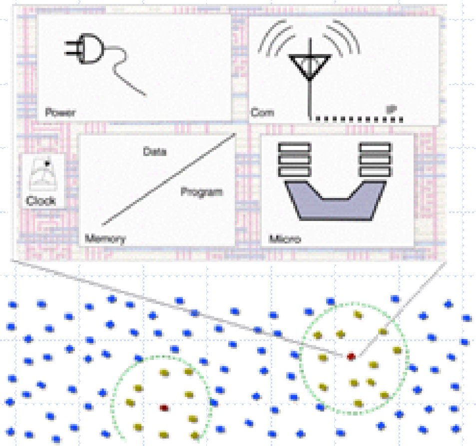
- Processes have only limited knowledge mostly about other processes in their neighborhood.
- Application is composed of local algorithms.
- Less inter-process dependencies, e.g., not everyone needs to know when a process dies.
- Peer-to-peer communication with overlapping neighborhoods promotes scalability.



Paintable Computing at MIT

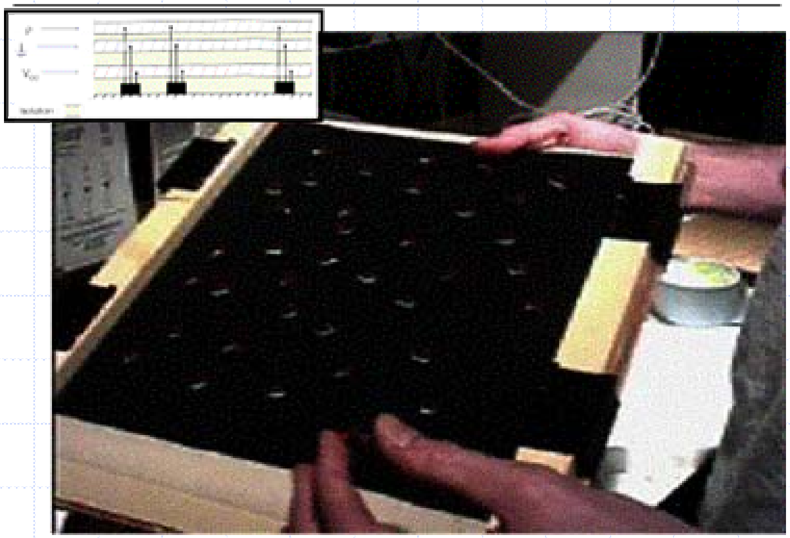
- In the future embedded computers with a radio device get as small as a pigment.
- Supercomputers can be easily assembled by painting a wall of embedded computers.
- Applications are driven by cellular algorithms.

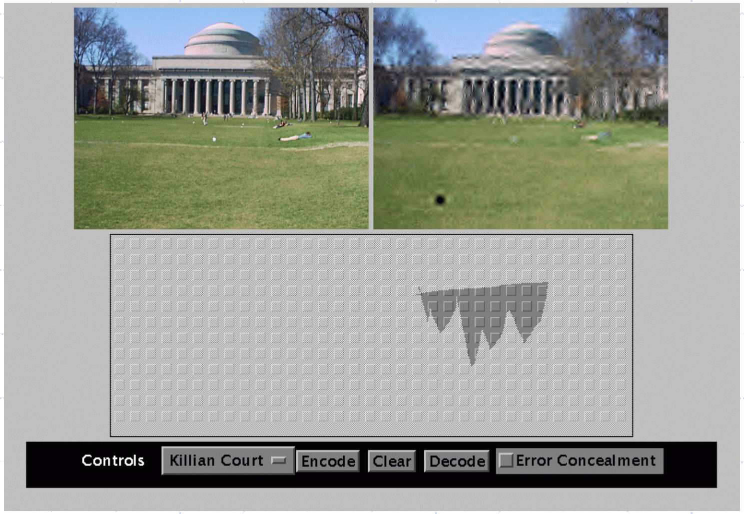




Paintable Computing at MIT

- 2002 Ph.D. Thesis using a pushpin board.
- Applications:
  - Distributed audio stream storage.
  - Fault-tolerant holistic data storage.

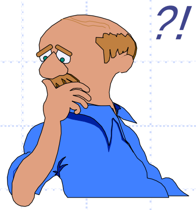




ORNL/IBM Collaboration

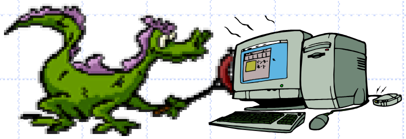
- Development of biology and material science applications for super-scale systems.
- Exploration of super-scalable algorithms.
  - Natural fault-tolerance.
  - Scale invariance.
- Focus on test and demonstration tool.

- Get scientists to think about scalability and fault-tolerance in super-scale systems!



ORNL Research Group

- Al Geist (PI)
- Christian Engelmann (simulator)
- Kasidit Chanchio (global max problem)
- Ryan Adamson (async. multigrid)
- Bill Shelton (LSMS port)
- Pratul Agarwal (MD port)



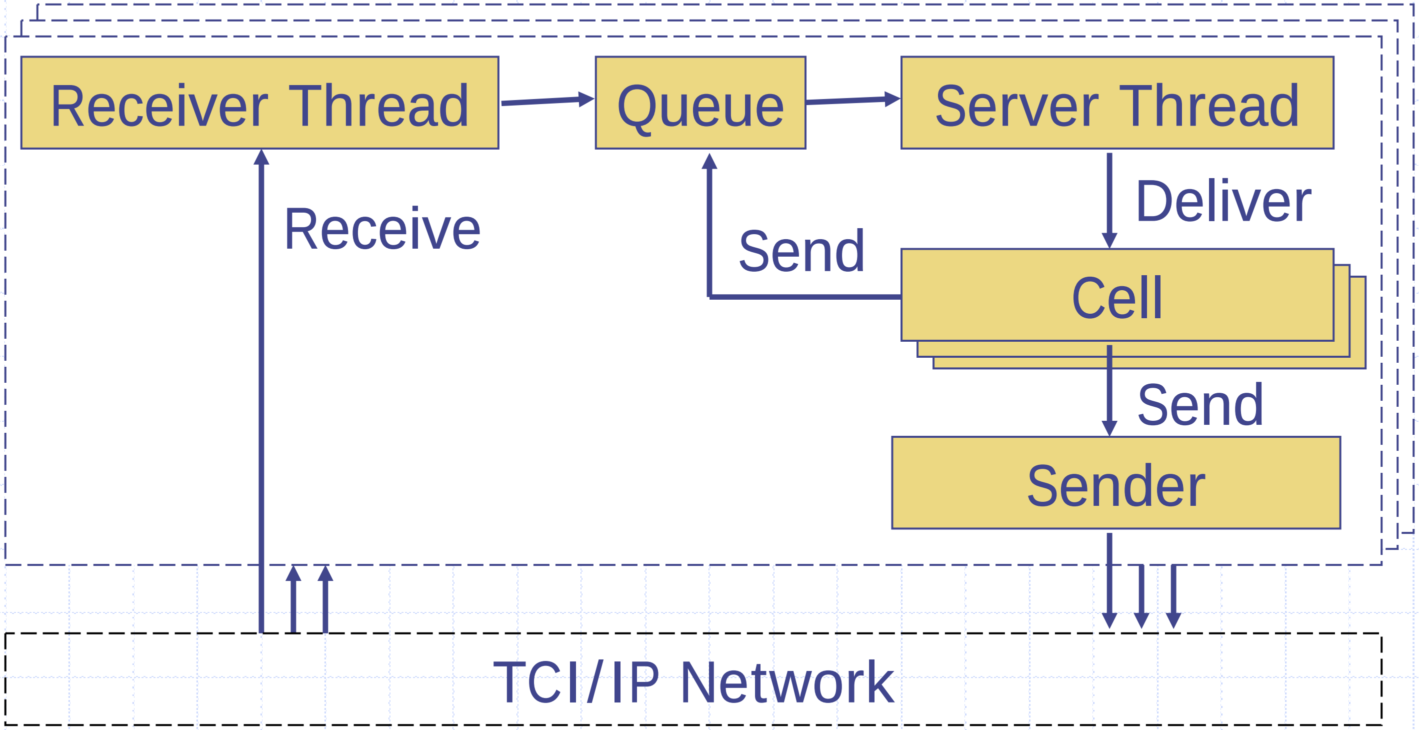
BlueGene\L Emulators

- IBM Research:
  - Processor emulation with OS in a Linux process.
- Caltech:
  - MPI trace file analysis for performance prediction.
- UIUC:
  - Object-oriented message driven emulation of logical system architecture in Converse/Charm++.
  - Adaptive MPI emulation on top of Charm++.
  - Scalability and performance issues in prototypes.
  - Emulation fixed on BlueGene\L architecture.

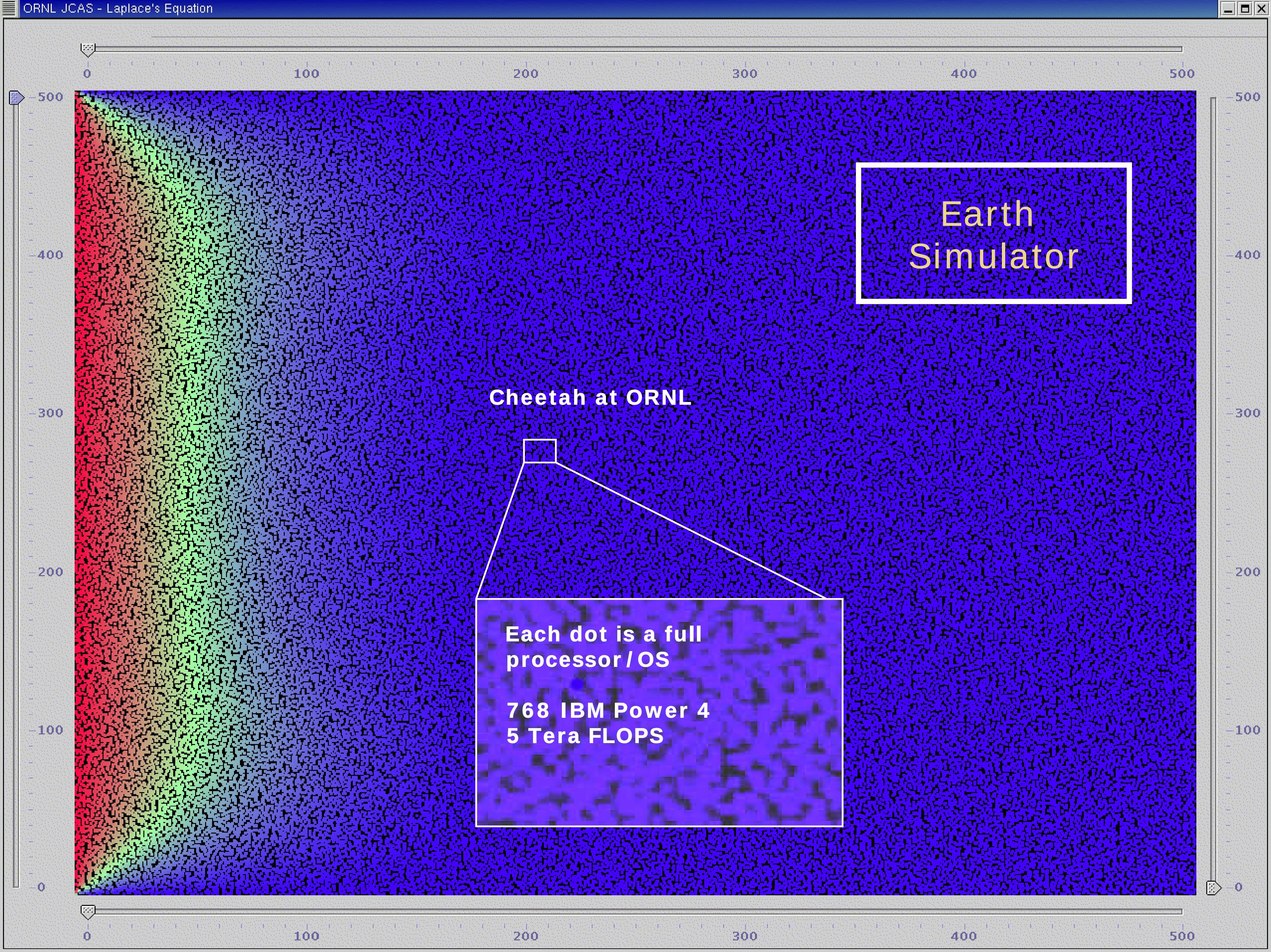
Cellular Architecture Simulator

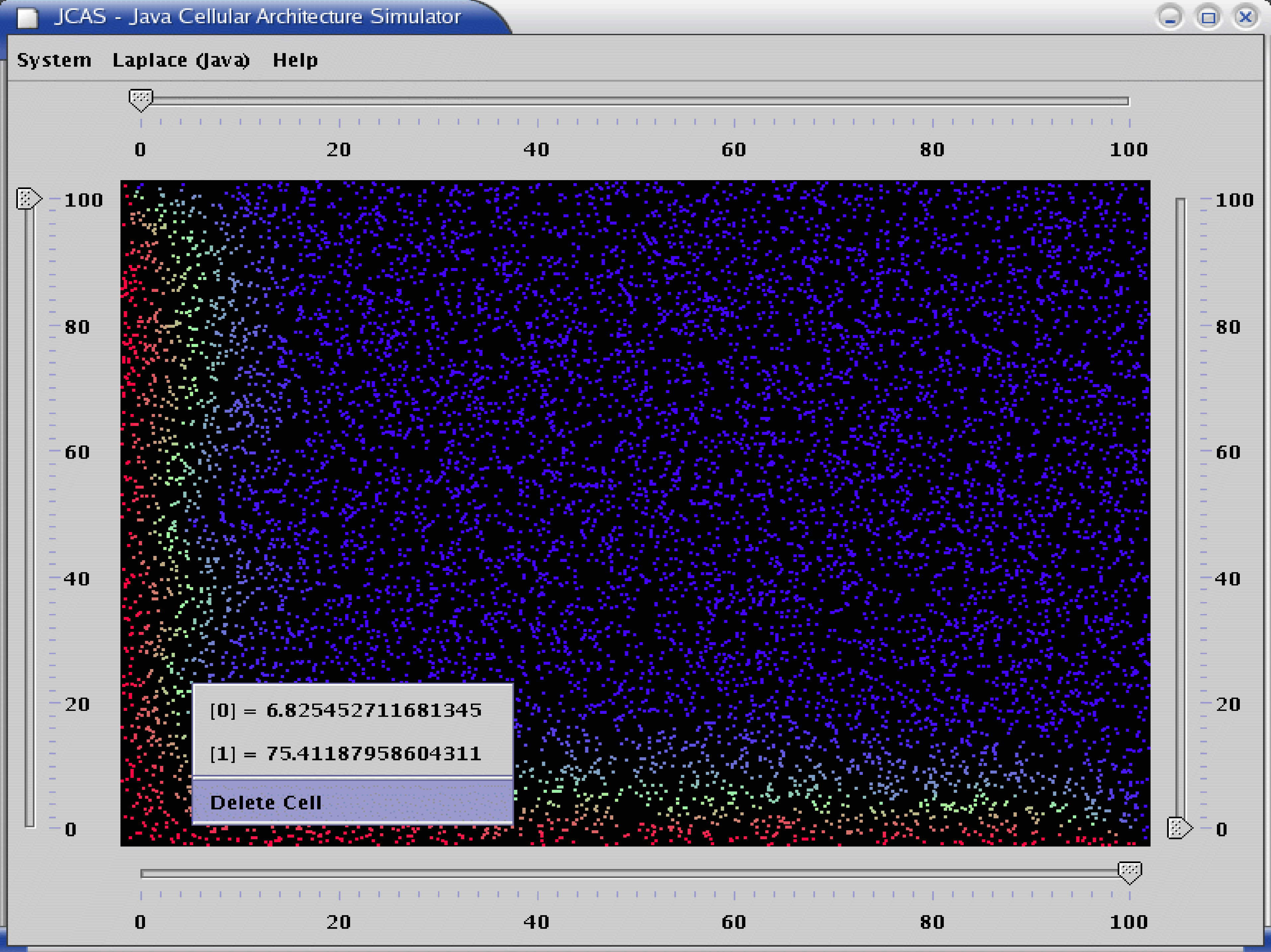
- Developed at ORNL in Java with native C and Fortran application support using JNI.
- Runs as standalone or distributed application.
- Lightweight framework simulates up to 1,000,000 processes on 9 real processors.
- Standard and experimental networks:
  - Multi-dimensional mesh/torus.
  - Nearest/Random neighbors.
- Message driven simulation is not in real-time.
- Primitive fault-tolerant MPI support.

Cellular Architecture Simulator



- Every cell has its own code, memory and neighbors list.
- Server hosts cells and initiates the context switch.
- Cells communicate asynchronously using messages.


ORNL JCAS - Laplace's Equation
Earth Simulator
Cheetah at ORNL
Each dot is a full processor / OS
768 IBM Power 4
5 Tera FLOPS

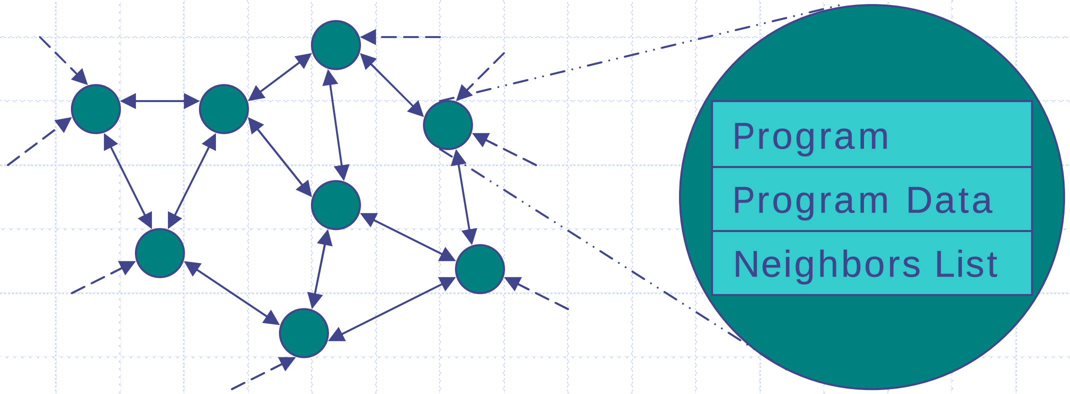

JCAS - Java Cellular Architecture Simulator
System
Laplace (Java)
Help
0
20
40
60
80
100
[0] = 6.825452711681345
[1] = 75.41187958604311
Delete Cell

Super-scalable Algorithms

- Extending the cellular algorithms theory to real world scientific applications.
- Exploring super-scale properties:
  - Scale invariance
  - Natural fault-tolerance
- Gaining experience in programming models for 100,000-processor machines.

Scale invariance

- Linear scalability.
- Peer-to-peer communication patterns are based on a small set of neighbor processes.
- Neighbors are random, far away or nearby.
- Global application state is composed of many interdependent local neighborhood states.



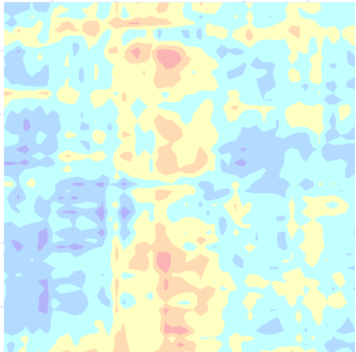
Natural Fault-tolerance

- Ability to get the correct answer despite task failures and without checkpointing.
- May involve redundant computation.
- 0,1% failure rate (100 of 100,000 processors) is still acceptable with 0,5% redundancy.
- Failures detected by hardware and ignored or accepted by neighbor processes.
- Failed processes may be restarted by “inserting” new ones at anytime.

Researched Algorithms

- Local information exchange:
  - Local peer-to-peer updates of values.
  - Mesh-free chaotic relaxation (Laplace/Poisson).
  - Finite difference/element methods.
  - Dynamic adaptive refinement at runtime.
  - Asynchronous multi-grid with controlled or independent updates between different layers.
- Global information exchange:
  - Global peer-to-peer broadcasts of values.
  - Global maximum/optimum search.

Ported Applications



- Material Science:
  - Magnetism simulation using the locally self-consistent multiple scattering (LSMS) method for understanding the interactions between electrons and atoms in magnetic materials (Bill Shelton).
- Computational Biology:
  - Molecular dynamics (MD) simulation of biological molecules (DNA sequences) for understanding the protein-DNA interactions (Pratul Agarwal).

Observations

- Partially non-deterministic algorithm behavior.
- Unpredictable application running time.
- Chaotic relaxation does not always converge.
- No exact replay without full message trace.
- Communication bound algorithms that require high point-to-point bandwidth.
- Asynchronous message driven programming model similar to discrete event simulations.
- Message queues with overwrite.

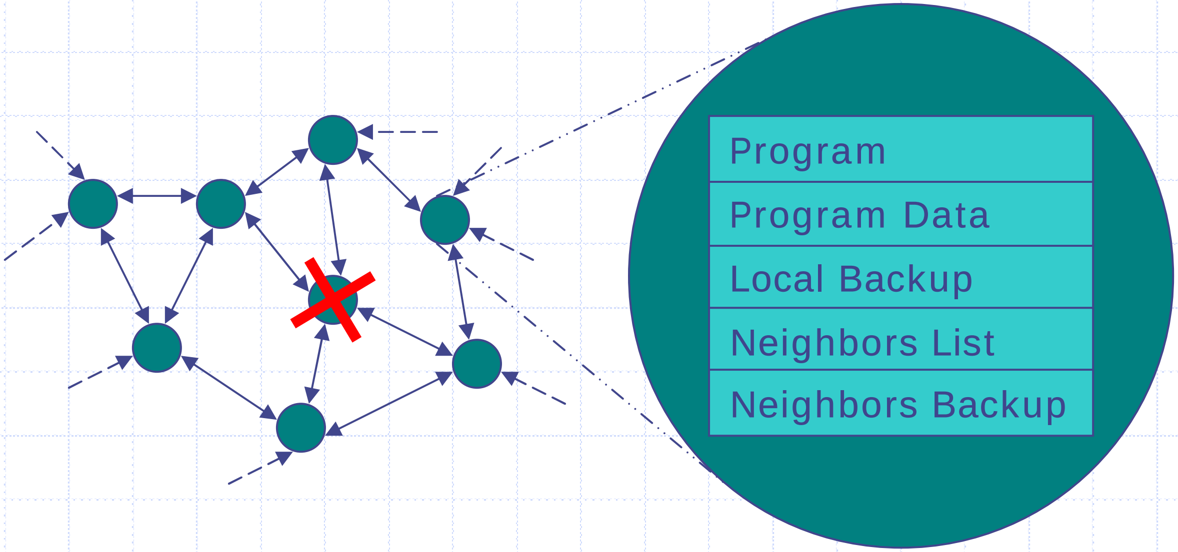
Super-scalable Fault-tolerance

- For non-naturally fault tolerant algorithms.
- Does it makes sense to restart all 100,000 processors because one failed?
- The mean time between failures is likely to be a few hours or just a few minutes.
- Traditional centralized checkpointing is limited by bandwidth (bottleneck).

- The failure rate is going to outrun the recovery and the checkpointing rate.

Diskless Checkpointing

- Decentralized peer-to-peer checkpointing.
- Processors hold backups of neighbors.
- Local checkpoint and restart algorithm.
- Coordination of local checkpoints.



Diskless Checkpointing

- In case of a failure:
  - Rollback to local memory backup if necessary.
  - Restart from remote memory backup.
- Encoding semantics, such as RAID, trade off storage size vs. degree of fault tolerance.
- Very infrequent checkpointing to central stable storage (disk/tape).
- Checkpoint and application processes may be the same or different.
- Possible OS support via library/service.

Choosing Neighbors

- Physically near neighbors:
  - Low latency, fast backup and recovery.
- Physically far neighbors:
  - Recoverable multiprocessor node failures.
- Random neighbors:
  - Medium latency and bandwidth.
  - Acceptable backup and recovery time.
- Optimum: Pseudorandom neighbors based on system communication infrastructure.

Backup Coordination

- All checkpoints need to be consistent with the global application state.
- Local states and in-flight messages.
- No coordination for checkpoints with no communication since last or since start.
- Coordination techniques:
  - Global synchronization
  - Local synchronization

Global Synchronization

- Global application snapshot (e.g., barrier) at stable global application state.
- Synchronous backup of all local states.
- Easy to implement.
- Synchronizes complete application.
- Preferred method for communication intensive applications.

Local Synchronization

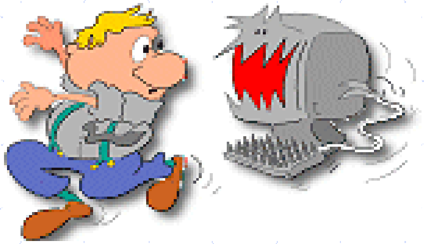
- Asynchronous backup of local state and in-flight messages (message logging).
- Acknowledgements for messages to keep accurate records of in-flight messages.
- Additional local group communication.
- Different methods to retrieve missed messages from neighbors.
- More complicated to implement.
- Preferred method for less communication intensive applications.

Observations

- Diskless peer-to-peer checkpointing on super-scale architectures is possible.
- Synchronization methods have different strengths and weaknesses.
- Timing, latency and bandwidth data impossible to obtain from simulator.
- Real-time tests with different applications are needed for further discussion.
- Final real-world implementation requires super-scalable FT-MPI or PVM.

Conclusions

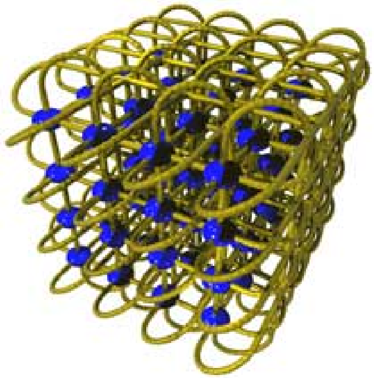
- Super-scale systems with 100,000 and more processors become reality very soon.
- Super-scalable algorithms that are scale invariant and naturally fault-tolerant do exist.
- Diskless peer-to-peer checkpointing provides an alternative to natural fault-tolerance.
- A lot of research still needs to be done.



Ideas for the Future

- Research in OS and/or middleware supported super-scale diskless checkpointing.
- Development of super-scalable fault-tolerant MPI implementation with localized recovery.
- Development of super-scalable algorithms for specific applications in computational biology, material science, climate research ...

January, 2003



Super-scalable Algorithms

Next Generation Supercomputing on 100,000 and more Processors

Christian Engelmann