HIVE: Fault Containment for Shared-Memory Multiprocessors
J. Chapin, M. Rosenblum, S. Devine, T. Lahiri, D. Teodosiu, A. Gupta

CSE 598C
Presented by: Sandra Rueda
The Problem

- O.S. for managing FLASH architecture (large shared-memory multiprocessor)

- Set of nodes connected in a mesh
- NUMA
Hive: Main Goals

- Memory Sharing: improving Performance

- Possible Failures:
  - A faulty node makes that node’s memory inaccessible
  - A faulty node returns wrong values for reads
  - Software failures may corrupt other node’s memory
Main Goals

- **Fault Containment:** Hardware or software faults are confined to the cell where they occurred, as a consequence just that cell crashes.

- **Scalability:**
  - Few resources are shared among different cells.
  - More processors → more cells → more parallelism
Multicellular architecture: processors are grouped into cells. An independent kernel manages each cell (UNIX SVR4).

Cell Organization:

Memory Organization:
- Cell 0
- Cell 1
- Cell n

Global Address Space

Local Address Space
Fault Containment (1)

Failure Sources

- Sources and control methods:
  - Message exchange (RPC):
    - timeout + message check
  - Remote reads:
    - careful_reference + message check
  - Remote writes:
    - Internal data: firewall
    - User level data:
      - Protection of local space
      - Preemptive discard
Fault Containment (2)
Control Methods

- Careful_reference protocol prevents errors from causing a kernel panic.
  - Save context
  - Check the memory range belongs to the expected cell
  - Copy data values
  - Check every remote data structure
  - Careful_off
Fault Containment (3)
Control Methods

- The Firewall controls which processors are allowed to modify each region of main memory.
  - Only the local processor can change firewall bits.
  - Rights are assigned to:
    - First process that requests a writable mapping to the page.
    - All the processors in a cell.
- Preemptive Discard (recovery)
Fault Containment (4) Detection

- Detection of a failure:
  - RPC request times out
  - Memory reading operation causes a bus error
  - Periodic updating of a shared location fails
  - Data fails consistency check

- When a failure is detected then an agreement protocol is run among other cells
Fault Containment (5)

Recovery

- **First Phase:**
  - Each cell flushes its TLB and remove any remote mapping.

- **Second Phase:**
  - At the end of the first phase there is no pending remote access, so it is possible to revoke firewall write permissions.
  - The virtual memory subsystem detects pages that were writable by a failed cell and notifies to the file system.
Fault Containment (6)

Recovery

- Preemptive Discard:
  - It is possible for a process to fetch stale data from disk after a recovery
  - Only processes that opened a file before a failure will receive I/O errors. It is implemented with a generation number, mismatches about the number will generate an error.
Memory Sharing (1)

- Two types of memory sharing:
  - logical level: a process on a cell maps a data page from another cell into its address space

```
+----------------------------------+
| cell i                           |
| pfdat table                     |
| exp                             |
| mem pages                       |
+----------------------------------+
+----------------------------------+
| mem pages                       |
| imp                             |
| pfdat table                     |
+----------------------------------+
```
Memory Sharing (2)

- Two types of memory sharing:
  - physical level: one cell transfers control over a page frame to another
Memory Sharing (3)

- **WAX:**
  - It is a user level process that may have access to all cells. In this way it is able to consolidate a global view of the system.
  - Some decisions are made based on the global view. For instance processes priorities.
RPC: Optimization

- Some times cells exchange information via RPC
- FLASH architecture includes hardware support to minimize RPC latency
- The mechanism is based on the cache-line delivery mechanism used by the cache coherency protocol (SIPS: Short Interprocessor Send Facility)
  - Primitive is reliable
  - No message fragmentation
Experimental Results

- At the time of the paper
  - Hive was a prototype
  - FLASH hardware was not available yet
  - Authors used SimOS
Simulation Environment

- **Hardware**
  - 4 processors MIPS 200 MHz
  - memory 128 MB
  - 4 disk controllers, each with one attached disk
  - 4 ethernet interfaces
  - 4 consoles

- **Hive**
  - 4 cells
  - each cell: 1 processor, 32 MB memory, 1 interface, 1 disk
Simulation Environment

- Memory Hierarchy (per processor):
  - Instruction cache: 32 K, two-way-associative
  - Primary data cache: 32 K, two-way-associative
  - Secondary unified cache: 1 MB, two-way-associative
  - Given miss penalty
- Given SIPS latency
- Given interrupt latency
- Given disk latency
- Some values are based on other models
Simulation Environment

- Performance Tests
  - Expected workloads (scientific application, parallel application)
  - Times for IRIX 5.2 (reference)
  - Different configurations: one, two, four cells
  - Conclusion: The partition into cells has little effect on performance, and it allows fault containment
Simulation Environment

- Fault Injection Tests
  - Difficult to predict the reliability of a complex system
  - Fault injection tests are used to detect if reliability mechanisms are working properly
  - Authors chose to inject failures in situations where it seemed that a fault in one cell could corrupt another
  - They checked files after recovery to detect data corruption
  - The simulator allowed them to recreate scenarios from a specific checkpoint
Conclusion
Simulation Environment

- Advantages [1]
  - Evaluation of hardware support
  - Evaluation of designed mechanisms
  - Evaluation of tradeoffs

- Problems [2]
  - Simulator Bugs
  - Omissions
  - Lack of Detail

- Key Features define if it is useful
References