Aakash Saini(2018CSM1001)

May 7, 2019

- **Observations:** This paper proposes the Hive prototype for the complete implementation of UNIX SVR4 which is targeted to run on Stanford FLASH multiprocessor. Hive focuses the following challenges: (1) *fault containment* by confining the effects of the faults in cell in which they occur (2) *Memory sharing* among the cells which is required to achieve application performance competitive to other multiprocessor OS and (3) *Single-system image* to present the standard SMP OS interface to the users and application. Hive prevent the wild writes by using special-purpose hardware and trusted software which uses mircokernel as the trusted base. Hives' architecture can be classified into two categories: fault containment part and resource sharing part. Fault containment at the hardware level is a hardware design problem, with requirements specified by the memory fault model that Hive relies on. *Message exchange, Remote reads* and *Remote Writes* restricts the damage of one cell to another cell for which **pages to protect are chosen** and **wild writes to unprotected pages**. **Resource sharing architecture** implements *Resource sharing mechanisms* and *Resource sharing policy*. Hive uses a two-part solution. First, cells monitor each other during normal operation with a number of heuristic checks. A failed check provides a hint that triggers recovery immediately.Second, consensus among the surviving cells is required to reboot a failed cell. When a hint alert is broadcast, all cells temporarily suspend processes running at user level and run a distributed agreement algorithm. Hive model a machine similar in performance to an SGI Challenge multiprocessor with four 200-MHz MIPS R4000 processors and a 700 nanosecond main memory access latency. Two workloads *pmake* and*raytrace* were used for the evaluation of performance of Hive.
- **Drawbacks:** The design of single-system image for the OS is only partially completed. Current prototype is not sufficient to demonstrate the fault containment in shared-memory multiprocessors. Differnce in latency between interrupt-level and qued RPCs affects the performance of Hive system due to the high context switching and synchronization cost.
- **Conclusions:** Hive improves the reliability of large-scale shared memory OS which are heavily used at the general purpose servers. Fault containment mechanism of Hive provides the memory isolation through a write protection hardware and software strategy. The multi structured of Hive provides it scalibility for the multiprocessors larger than the current systems. Hive architecture provides the environment of reliable benefits by synchronizing the actions of various cells. Load balancing and resource reallocation are designed to be driven by a user-level process and provide a global view of system.
- **Future work:** In future single-system image can be further extended and shared memory systems can be provided with the fault-containment. Hive can be further enhanced to lessen the context switches.