

NUMA  
on  
HP-UX



**i n v e n t**



Bjorn Helgaas  
Linux Development Lab  
24 March, 2001

Page 1

# NUMA Experiences in HP-UX

Bjorn Helgaas  
Linux Development Lab  
Ft. Collins, CO



# overview

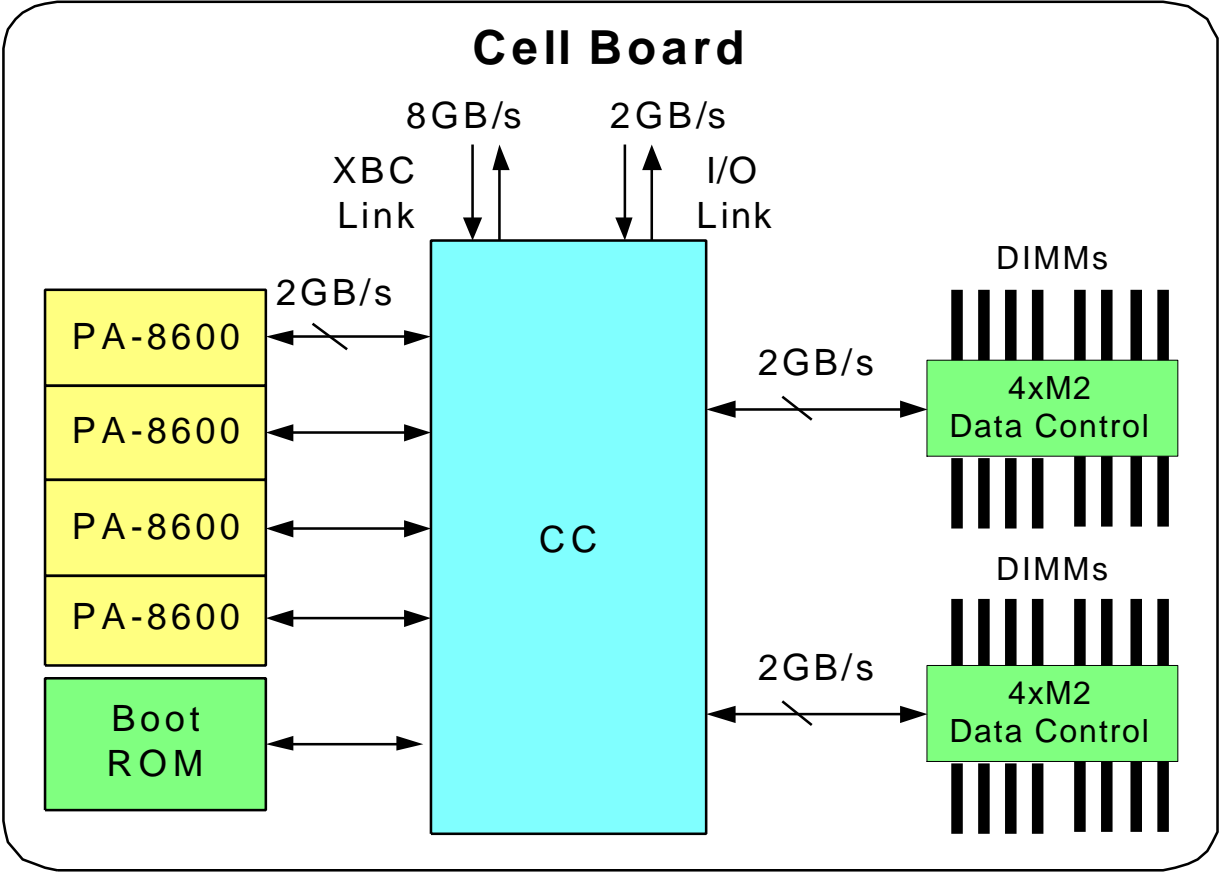
- superdome system architecture
  - single-cell SMP
  - multi-cell ccNUMA
- programming model
  - topology discovery
  - memory placement
  - process/thread placement
- experiences



NUMA  
on  
HP-UX

single-cell  
SMP

# superdome cell board

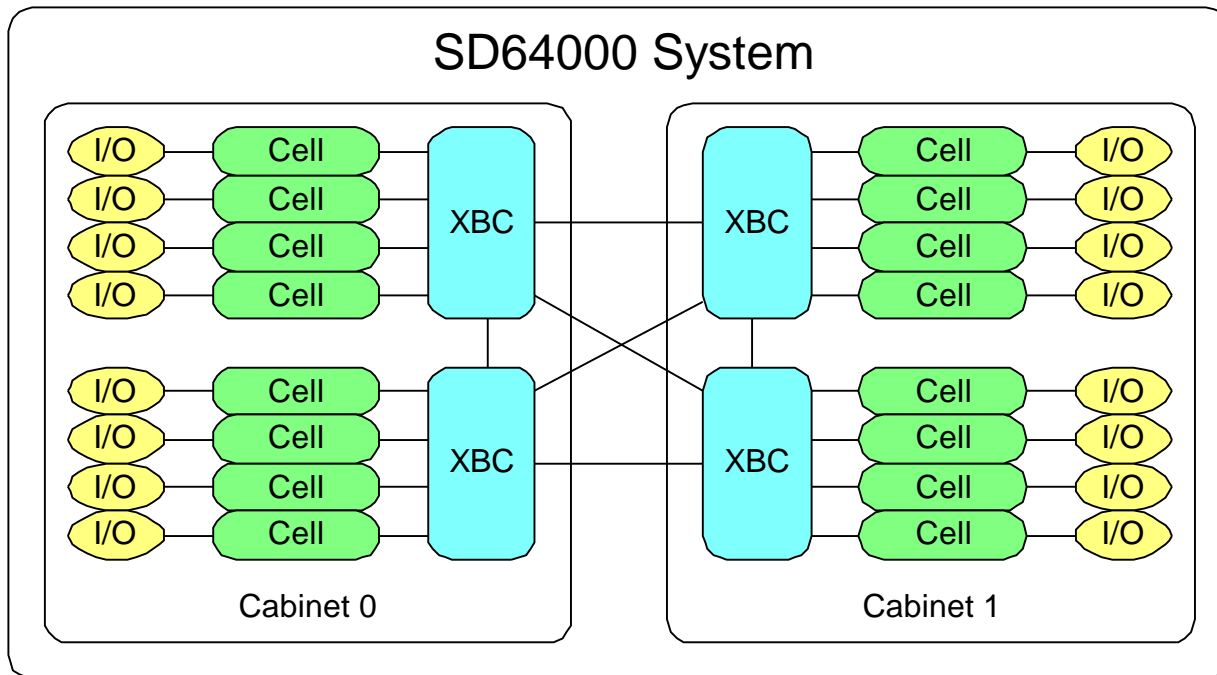


NUMA  
on  
HP-UX

multi-cell  
ccNUMA

# 64-way superdome system

- memory accessible by all CPUs & I/O
- both interleaved and cell-local memory



# programming model

- topology discovery
- locality domain management
- memory placement
- process/thread placement



# topology discovery

- enumerate CPUs in system  
mpctl(2): MPC\_GETNUMSPUS,  
MPC\_GETFIRSTSPU, MPC\_GETNEXTSPU
- enumerate “locality domains” in system  
mpctl(2): MPC\_GETNUMLDOMS,  
MPC\_GETFIRSTLDM, MPC\_GETNEXTLDM,  
MPC\_GETLDMSPUS
- map a CPU to a locality domain  
mpctl(2): MPC\_SPUTOLDOM
- no information about I/O topology
- no information about memory topology (i.e., size of local memory)

# locality domain management

- physical description only
- no abstraction
  - cf. IRIX “memory locality domain” with memory size N
- no aggregation of multiple locality domains
  - cf. IRIX “memory locality domain set” with affinity for devices

# memory placement

- default policy is first-touch
  - memory allocated from locality domain in which fault occurs
- can be overridden for mmap and shmget regions
  - mmap(2): MAP\_LOCAL
  - shmget(2): IPC\_MEM\_LOCAL
- no external influence (requires code change at mmap/shmget call site)

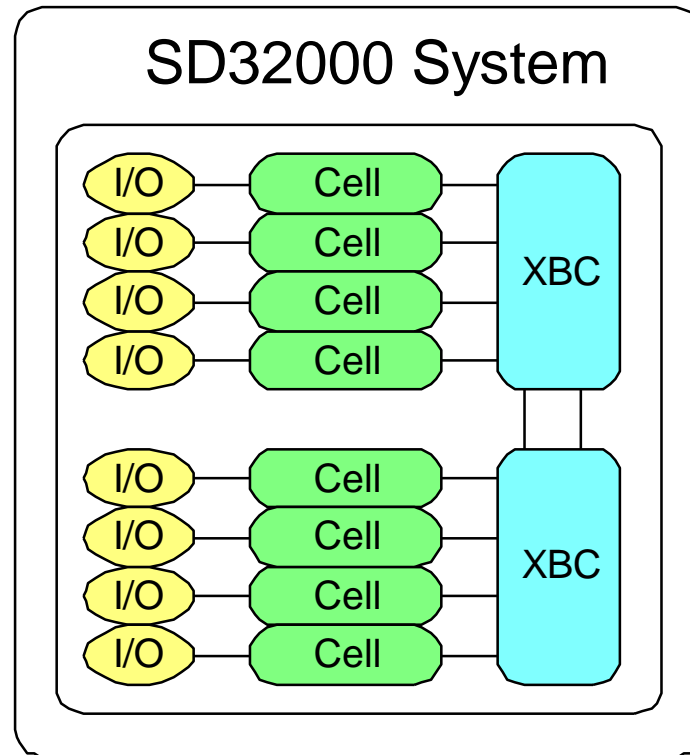
# process/thread placement

- default policy is NONE:
  - processes placed in least-loaded locality domain
  - threads placed to “fill” current locality domain, then spill into least-loaded domain
- other policies: round-robin (across localities), fill-first, packed, least-loaded
- locality domain bindings inherited by children
- set “internally” via mpctl(2)
- external control via mpsched(1)
  - mpsched -p <policy> command

# experiences

- insufficient memory placement support
  - can request memory in current locality domain, but not memory distributed across cells
- insufficient process placement support
  - large apps consist of many scripts and executables; impossible to “zero-in” on the important ones without code changes
- insufficient visibility and instrumentation
  - no visibility of actual memory distribution, usage of local/remote memory, or even size of local memory
- no I/O locality support
  - no device topology information, no way to place memory or processes close to device

# 32-way system



# I/O - 12-slot PCI chassis

