ORACLE®

Considerations for Thread Binding

Dave Dice

Oracle Labs

Bound threads

- When is binding appropriate ?
  - Dedicated system : router; embedded; HPC; known HW and SW
  - Exploration and sensitivity experiments
    - learning phase for algorithm development
    - Simplifies
- Reduces run-to-run variance
  - Fewer runs (time) needed to collect stable data
- Need to explore multiple placement policies
  - Maximum distribution over cores / sockets / caches
  - Maximum ”packing” density
  - Partial packing
  - etc

Bound threads : considerations

- Requires intimate knowledge of topology and CPUID mapping
  - Shared caches; meshes or ring position; NUMA; HT etc
- Not portable
- Not realistic or faithful to real-word environments
- Unfriendly to virtual machines
  - Virtualization attempts to distance app from HW
  - Binding works in the opposite direction
- Devops and deployment portability
- Mutually unaware processes can collide in binding decisions

Bound threads : considerations

- Impedes reproducibility
- Self deception is easy
  - Performance : seen ***frequently*** in academic papers
  - Correctness : Hierarchical CLH -- binding hid algorithm flaw!

Bound threads : considerations

- Impedes **Spectre Mitigation**
- OS tries to isolate unrelated threads
- Reduces signal for cache timing attacks

Bound threads : considerations

- Unfriendly to Asymmetric cores – heterogenous performance SMP
- **P**erformance-**E**nergy (Intel); Fire-Ice (M1); Big-Little (ARM); etc
- Intel’s “thread director”
- Static association between CPUID and performance
- Instead of dynamic performance : SMT; turbo; etc

Bound threads : considerations

- Intel QPI → UPI
- Shift to home-based snooping
- Linux 6.x maximum dispersion placement policy : threads onto NUMA nodes
- Self-binding via pthread_setaffinity() will “tear” thread
  - Away from stack
  - Away from TLS
  - Simple misses become more expensive : local ➞ remote misses
  - Impact on MCS performance is cache elements end up on wrong node !
- Need to place **data** and **threads** to bind effectively

Bound threads : considerations

- Scheduler makes thread placement decisions
  - Bound threads deny scheduler latitude to balance threads over resources
    - Migration
  - Global system state visible to scheduler
    - Unbound → global and dynamic placement decisions
    - Bound → local and static
  - Bound threads tantamount to -- I know better than the scheduler

Are you the omniscient Technoking ?

Bound threads : considerations

- Unbound default free-range threads :
  - harder problem but stronger result
  - Arguably better representative of the “real world” : commercial; cloud
  - Respond appropriately to ambient placement