Challenges in Operating System Design for Future Many-Core Systems

Tommaso Cucinotta
Real-Time Systems Laboratory
Scuola Superiore Sant'Anna
Pisa, Italy
General-Purpose Computing (GPC)

- General-Purpose Hardware
  - Limited parallelism degree (few cores era)
- OS provides useful services to applications, e.g.:
  - Hardware abstraction
  - (GP) Scheduling of resources (e.g., tasks on available CPUs)
    - Automatic separation between interactive and batch applications
  - (GP) Filesystem, I/O and networking
  - ...
- Applications mostly sequential (with a few exceptions)
  - Application-level programmers
  - OS-level (and kernel-level) programmers
High-Performance Computing (HPC)

- Specialized hardware
  - Vector machines, ...
  - Massive parallelism degree
- OS constitutes a “noise” (or “jitter”) to get rid of
  - Applications often optimized for underlying hardware
- Optimized distributed filesystems
- Application-specific distribution and scheduling logic
  - Assumption of availability of entire system: no need for caring about multiple applications multiplexed on the same system
- HPC programmers are experts of
  - parallel programming techniques
  - ...
Future Many-Core Systems

- Potentially suitable for both (high-end) GPC, CC and HPC
- Increasing need for a good OS-level support
  - data distribution and replication
  - workload distribution, load balancing and scheduling
  - management of complex memory hierarchies and incoherent shared memory segments
- Nowadays OSes unable to efficiently manage many cores
  - Monolithic kernels
  - Global in-kernel data structures (e.g., processes, file-system)
  - Global in-kernel synchronization spin-locks (e.g., Linux bkl)
  - Even fine-grained locking (e.g., object-level lock) inefficient
    - When thousands of cores may potentially compete
What has been proposed?
(for scalability in # of cores)

**Multikernel (and Barrelfish prototype)**
- One OS instance per-core
- Any sharing implemented by message-passing between different kernel instances

**Partitioning of cores (Corey OS, GenerOS, FOS)**
- Application cores
- Kernel/service cores
- For example, a system call becomes a RPC

**Application-level control of sharing (Corey OS)**
- Help the kernel understand what is likely to be accessed by multiple tasks and what cannot
  - Non-sharing default policy, sharing needs explicit actions
Real-Time Applications

**Time-sensitive applications**
- Throughput and/or latency constraints
- Computation times vary depending on data locality
- We don't want to design everything off-line
  - but we expect to have a proper run-time OS-level support
- Scheduler needs to be real-time aware
- Adaptivity plays a key role

**Real-Time Scheduling on Multi-Processors**
- Many open problems
- No known algorithm for efficient use of many CPUs
**Optimum/reasonable deployment of VSNs on PNs**

- Given computing/network/memory requirements
- Respecting end-to-end timing constraints

Diagram:

- Virtual Service Network
- Maximum response-time
- Computing/Memory Requirements
- Networking Requirements
- Physical Host
- Physical Subnet
- Physical Link
Scheduling Challenges

**Distributed Scheduling Infrastructure**
- **No centralized** scheduling decisions
- **Hierarchical management** of resources (and scheduling)
- What properties can we guarantee system-wide?

**Synchronization and IPC Mechanisms**
- More integration with scheduling mechanisms

**Application Programming Interface**
- What info do we need to expose to the scheduler?
  - Application-level DFG and dependencies?
  - (expected) Communication paradigms/patterns?
  - Timing constraints and (expected) latencies?
- What info can be **automatically inferred** by the kernel?
  - e.g., by (kernel-level) monitoring
More Scheduling Challenges

Heterogeneous Hardware

- Different CPUs have different performance
- How to properly take scheduling decisions?
- What goals to target?
  - maximize system throughput?
  - minimize maximum latency?
  - minimize energy consumption while keeping timing constraints?
  - ...?
- Adaptiveness: when to migrate tasks and how?
  - How to deal with the NUMA effect?
Thanks!