Evolution of parallel and cluster computing

Daniel Hagimont
daniel.hagimont@irit.fr

Toulouse Polytechnic National Institute

Is 128 bits address spaces a revolution ?

- Architectural Support for Single Address Space Operating Systems, E. Koldinger, J. Chase, S. Eggers, ASPLOS 1992
- *Consider that 40 bits can address a terabyte, two orders of magnitude beyond the primary and secondary storage capacity of all but the largest systems today, and that a 64-bit address space, consumed at a rate of 100 megabytes per second, would last five thousand years.*

- *128 bits - 100 terabytes/sec → 10^{17} years*

History of multi-processor systems

- Multi-processor SMP machines
- Multi-processor NUMA machines
- Multi-core processors
- Clusters of machines
  - SSI
  - DSM
  - SASOS
- More recent evolutions
  - Disaggregation
  - Multi-kernel
  - Specific runtimes
- Tendances

Multi-processor SMP machines

- SMP Architectures
  - Unique RAM shared by multiple processors
  - Uniform Memory Access (UMA)
  - In the 1990's (single-core processors at that time)
    - Sequent Balance 8000 (1984), IBM RS/6000 (1990), Sun SPARCserver 1000 (1989), DEC 7000 (Alpha, 1989)
  - Two ways to architecture these SMP
    - Rack mode: with a fast interconnect
      - Initially, and later for scalability
    - On motherboard: mutliple sockets, with interconnect
      - Later, more compact machines (Intel Xeon, 1998), (AMD Athlon, 2001)
  - OS (generally Unix) adapted for SMP support

Multi-processor NUMA machines

- NUMA Architectures
  - Each processor has a local RAM, but can access other processor's RAM (at a higher cost)
  - Non Uniform Memory Access (NUMA)
  - Because in SMP/UMA, interconnect and memory are bottlenecks
  - In the 1990's: Sequent NUMA-Q (Intel, 1992), SGI Origin 2000 (1996)
  - In the 2000's: support from Intel Xeon (2001) and AMD Opteron (2003)
  - Today's:
    - Intel Xeon family with QPI interconnect
    - AMD EPYC familly with Infinity Fabric
  - OS (generally Unix/Linux) adapted for NUMA support

Multi-core processors

- Multi-core architectures
  - In the 2000's
    - Dual-core: IBM Power4 (2001), Sun UltraSPARC IV (2003), Intel Pentium D (2005) , AMD Athlon 64 x2 (2005), Intel Core 2 Duo (2006)
    - Quad-core: AMD Phenom X4 (2007), Intel Core i7 with hyperthreading (2008)
  - From 2010 (for datacenters)
    - Intel Xeon Nehalem: 8 cores (2009)
    - AMD EPYC 7001: 32 cores (2017)
    - Intel Xeon Haswell: 18 cores (2014)
    - AMD EPYC 9004: 96 cores (2023)
    - Intel Xeon Sapphire Rapids: 56 cores (2023)
  - OS were already supporting multiple processors (because of SMP), except cache affinity

Clusters

- Clusters of machines
  - Machines interconnected with a LAN
  - In the 1990's
    - Clusters for High Availability or load balancing
      - Main applications: Web Servers (e.g. Apache HTTP Server) and databases (e.g; Oracle Parallel Server)
      - VAXCluster (DEC, 1990), Microsoft Cluster Server (1996)
    - HPC clusters
      - Main applications: scientific, simulations
      - Mainly based on MPI
      - Beowulf cluster (Lawrence Berkeley National Laboratory, 1994)
    - Only a set of machines exploited by specific applications

Clusters: DSM and SSI

- Distributed Shared Memory (DSM) Systems
  - Reproducing Unix shared memory (between processes, but cluster-wide)
- Attempts to manage a Single System Image (SSI)
  - Very early (from 1980)
  - Mosix (1982), Kerrighed (1998)
  - Manage process migration, maintaining sockets or shared memory segments consistency
  - Processes are transparently scheduled on several machines, but this is not like a SMP or (rather) NUMA machine, no distributed process
- Providing a DSM exploited by applications
  - A specific programming model for parallel applications
  - Many consistency models were studied
  - Ivy (Yale, 1989), Clouds (Georgia Tech), Munin (Utah, 1990), TreadMarks (Rice, 1991), Midway (UW, 1995)
  - Neither a SMP, NUMA nor SSI, an alternative to message passing

Clusters: SASOS

- Single Address Space Operating Systems (SASOS)
  - All allocated virtual addresses are unique cluster-wide
  - Independent from protection domains
  - Unique addresses can be exchanges between processes in the cluster or stored on disk (persistent object names)
  - Advantage: no need for any name translation (!!!)
    - e.g., pointer swizzling
    - e.g., passing a linked list as parameter of a RPC
  - Opal (University of Whasingtion, 1992)
- Remarks
  - This was motivated by the advent of 64 bit processors (*consumed at a rate of 100 Mb/s, a 64-bit address space would last for 5000 years*)
  - Somewhere, other DSM are relying on a SAS within the DSM

Clusters: issues with SAS

- SAS relies on large virtual addresses
- SAS is for naming and binding (only)
  - Same object names everywhere
  - Memory is potentially shared (depending on protection domains)
    - Not page-level only : false sharing (regarding consistency and protection)
    - Copy and consistency protocols
- Problem of large object names (virtual addresses)
  - Increasing object names' size
    - Increases size in memory and storage
  - Schemes with relative names were proposed in the 1990's
    - In storage
      - Relative names within a local space (shorter, e.g. 32 bits)
      - Fowarders when pointing to an object outside the local space (space name + relative name)
      - Reducing size in storage
    - Mneme (Amherst, 1990)

Disaggregated architectures

- Disaggregated architectures
  - Hardware resources (such as processors, memory, storage, and networking) are physically separated
  - High speed interconnect
  - Can be scaled, managed, and allocated independently
  - Rather than being integrated into a single monolithic system (motherboard)
  - This is already the case with SAN/NAS (storage)
- Intel Rack Scale Architecture – 2015
- Compute Express Link (CXL) - 2019
  - Open standard interconnect for high-speed CPU-to-device and CPU-to-memory connections
  - Supported by Intel and AMD and adopted by Amazon, Google
- What about the OS ?
  - Here, the rack becomes a single SMP with a SSI
  - LegoOS proposed SplitKernel (2018)

Multi-kernel

- BarrelFish (ETH Zurich, 2009)
  - Today's OS (on centralized machines) rely on consistent shared memory (of the operating system, between cores)
  - Rather rely on message passing between multiple kernels (one per core)
    - Less complexity when hardware is heterogeneous
    - Messages scale better compared to shared memory
  - Implement an OS as a distributed system, a set of cooperating kernels
- Can be applied to NUMA or disaggregated architectures

Specific runtimes

- Instead of cluster integration at the OS level (SSI)
  - User-level application specific runtimes for managing clusters
  - Only clusters composed of independent machines
- Exemples
  - MPI
  - Hadoop/Spark
  - Kubernetes
  - NB: a DSM system is such a example (not SSI)

Synthesis/Questions

- Tendances
  - Generalisation of NUMA multi-core servers
  - Disagregated architectures
  - Both for scalability and flexibility
  - Specific runtimes for very large scale systems
- Questions
  - Do we need a SSI for scaled multiprocessors ?
    - NUMA: instead of adapting Linux, implementing a multi-kernel ?
    - Diaggregated architectures: is split-kernel a kind of multi-kernel ?
  - Do we need a SSI for very large scale infrastructures ?
    - Not really, provided by specific runtimes
  - Do we need SASOS ?
    - Can be used at the OS level (but not exploited up to now)
    - Could be valuable in specific runtimes
  - Is shared memory the right paradigm ?
    - For implementing the OS: multi-kernel says no
    - For developping applications: depends