MxKernel

Rethinking the System Software Architecture for Multicore and Manycore Computers

http://mxkernel.org

Olaf Spinczyk

olaf.spinczyk@tu-dortmund.de
http://ess.cs.tu-dortmund.de/~os

in cooperation with Jens Teubner, TU Dortmund

Embedded System Software Group
Computer Science 12, TU Dortmund
Outline

- Project context: SPP 2037
- Problems with traditional OS abstractions
- Manycore programming in other domains
- Manycore OS: State-of-the-art
- The MxKernel software architecture
- Preliminary results
- Next steps
Outline

- Project context: SPP 2037
- Problems with traditional OS abstractions
- Manycore programming in other domains
- Manycore OS: State-of-the-art
- The MxKernel software architecture
- Preliminary results
- Next steps
SPP 2037: A DFG Priority Programme

- „Scalable Data Management for Future Hardware“
- 10 projects with 1-2 PIs each
  - Scalable Data Management in the Presence of High-Speed Networks (TU Darmstadt)
  - Interactive Big Data Exploration on Modern Hardware (TU Munich)
  - Query Compilation for the Heterogeneous Many Core Age (TU Berlin)
  - ReProVide: Query Optimisation and Near-Data Processing on Reconfigurable SoCs for Big Data Analysis (University Erlangen-Nuremberg)
  - Adaptive Data Mgmt. in Evolving Heterogeneous Hardware/Software Systems (Uni Magdeburg)
  - Distributed, fault-tolerant in-place consensus sequence on innovative hardware as a building block for data management (ZIB Berlin)
  - High-Performance Event Processing on Modern Hardware (Uni Marburg)
  - Transactional Stream Processing on Non-Volatile Memory (TU Ilmenau)
  - MxKernel: A Bare-Metal Runtime System for Database Operations on Heterogeneous Many-Core Hardware (TU Dortmund)

Manycore systems, GPU/FPGA accelerators, non-volatile memory, secure enclaves, RDMA, ...
Outline

- Project context: SPP 2037
- Problems with traditional OS abstractions
  - Manycore programming in other domains
  - Manycore OS: State-of-the-art
  - The MxKernel software architecture
  - Preliminary results
  - Next steps
Flaws of Traditional Operating Systems
... in the context of modern multicore and manycore systems

**Past**

**Hardware**
- Single CPU
- User/supervisor mode
- Uniform physical memory
- MMU: Virtual memory
- Global I/O controllers

**Future**

**Hardware**
- Many CPU cores
- Heterogeneous cores
- Complex NUMA architecture
- Non-volatile memory
- Non-uniform I/O architecture
- Voltage/frequency islands

**OS Support**
- CPU Multiplexing
- Monolithic architecture
- Huge virtual address spaces
- Lock-based synchronization

?
Flaws of Traditional Operating Systems

- Example: Linux memory allocation benchmark [1]
Outline

- Project context: SPP 2037
- Problems with traditional OS abstractions
- Manycore programming in other domains
- Manycore OS: State-of-the-art
- The MxKernel software architecture
- Preliminary results
- Next steps
Manycore Programming: Intel® TBB [2]

- Instead of threads: “Task-based Programming”
  - Fine-grained units of work: functions, functors, or C++ lambdas
  - Light weight: No separate stack, register set, etc.
  - Dependency graph: Parallel computation represented as a tree
- Task scheduler
  - Efficiently executes tasks from double ended queues
  - Automatic load balancing
- Problems
  - Inefficient if tasks perform blocking operations
  - Tasks must be synchronized by classic mechanisms
Manycore Programming: HyPer Morsels [3]

- Instead of threads: “Morsel-driven query execution”
  - Small DB operator pipelines, JIT compiled
  - Small chunks of input data
  - Input and output are NUMA-local

- Scheduler (in user space)
  - Fixed number of pinned threads
  - Load balancing by work stealing
  - Excellent scalability: 30x performance on 32 core system

- Problems
  - Special purpose solution
  - Does not re-use OS features

Figure 1: Idea of morsel-driven parallelism: $R \Join_A S \Join_B T$
Outline

● Project context: SPP 2037
● Problems with traditional OS abstractions
● Manycore programming in other domains

● Manycore OS: State-of-the-art

● The MxKernel software architecture
● Preliminary results
● Next steps
Manycore OS: State-of-the-Art

- Barrelfish [4]
  - Multikernel architecture

- fos [1]
  - Microkernel
  - Server threads (or “fleets”)

- Tesselation [5]
  - Cell concept
  - Gang scheduling

→ Still using threads. Optimizations done by app. programmer.
### Manycore OS: Apple’s GCD Kernel Support

- "Grand Central Dispatch"
  - Resembles TBB, but MacOS provides kernel-level support

<table>
<thead>
<tr>
<th>Serial dispatch queue</th>
<th>Dispatch source</th>
<th>Queue hierarchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>appl. threads</td>
<td>async. event</td>
<td>Q1</td>
</tr>
<tr>
<td>queue</td>
<td></td>
<td>Q2</td>
</tr>
<tr>
<td>worker thread</td>
<td></td>
<td>Q3</td>
</tr>
</tbody>
</table>

- Implicit serialization
- Worker thread creation on demand
- Seamless I/O integration
- Automatic triggering of success/failure handler
- Restricted number of threads
- Guaranteed partial order

### Problems
- Context switches for simple queue operations
  - Necessary to avoid priority inversion (task vs. thread priorities)
- No clean layer structure in the kernel
Outline

- Project context: SPP 2037
- Problems with traditional OS abstractions
- Manycore programming in other domains
- Manycore OS: State-of-the-art
- **The MxKernel software architecture**
- Preliminary results
- Next steps
The MxKernel: Key Features

- **Elastic cells**
  - Provide *spatial* isolation of applications and global OS services (based on priorities)
  - Support optimized mapping and performance isolation
  - Contain not only CPU cores, but also FPGA and GPU resources as well as NICs, memory, etc.
  - Are fully aware of assigned physical resources

- **Task-based programming model**
  - Simplifies development of parallel applications
  - Helps to avoid lock-based synchronization
  - Supports automatic load balancing, optimized task placement, and cell elasticity
  - Also suitable for distributed memory systems and heterogeneous computing

- **Global and local OS services**
  - Implemented on top of the task-based interface
The MxKernel: Architecture

**FPGA**
- brick (static)
- legacy OS & applications

**CPU cores**
- system software or application (task-based)
- MxTasking
- Resource model
- +/- cores
- +/- memory
- Current load or QoS per cell

**GPU**
- MxTasking
- Resource model
- +/- NICs
- +/- storage
- ... (ellipses indicate additional resources)

**MxVisor**
- monitoring
- adaptation
- manycore resource management strategies

**MxOS**
- device drivers
- OS services, e.g. network protocols, filesystems, etc.
- threads for legacy code
- family-based design

**MxTasking**
- task-based API
- elastic
- maintains topology view
- topology-aware optimizations (e.g. NUMA)
- fine-grained application-specific mapping decisions
- exploit heterogeneous computing resources

**MxVisor**
- isolation of cells
- priorities of applications
- optimized app-to-core mapping (NUMA-aware)
- power management
- anti-aging
- fault tolerance, e.g. app replication, handling damaged components
The MxKernel: Vision for OS/DBMS Relation

- M. Stonebraker, 1981: “Current DBMSs usually provide their own and make little or no use of those [services] offered by the operating system. [...] A DBMS would prefer a small efficient operating system with only desired services. Of those currently available, the so-called real-time operating systems which efficiently provide minimal facilities come closest to this ideal. On the other hand, most general-purpose operating systems offer all things to all people at much higher overhead. It is our hope that future operating systems will be able to provide both sets of services in one environment” [6]

- MxKernel (MxVisor+MxTasking)
  - focus on managing resources of modern many-core hardware
  - minimal base for other system software

- MxOS and MxDB
  - will be based on the task model
  - flexible composition in arbitrary elastic cells
  - Decomposed for mutual re-use
Outline

- Project context: SPP 2037
- Problems with traditional OS abstractions
- Manycore programming in other domains
- Manycore OS: State-of-the-art
- The MxKernel software architecture
  - Preliminary results
- Next steps
Preliminary Results

- Calculation of the Mandelbrot Set

**Mandelbrot Linux–PTHreads (4000x3000)**

Best Speedup: 27x @p125, 56 cores

- 80.9e+06 cycles

**Mandelbrot MxTasks (4000x3000)**

Best Speedup: 46x @p125, 60 cores

- 40.1e+06 cycles
Outline

- Project context: SPP 2037
- Problems with traditional OS abstractions
- Manycore programming in other domains
- Manycore OS: State-of-the-art
- The MxKernel software architecture
- Preliminary results
- Next steps
Next Steps

- Systematically explore the design space
  - Interfaces
  - Task metadata
  - Algorithms
- Database benchmarks
  - B-link tree, in-memory transactions
- Heterogeneous systems
  - Task variants for GPU and FPGA
- Power Management
References (1)


References (2)
