A MICROKERNEL-BASED OPERATING SYSTEM
FOR EXASCALE COMPUTING

Amnon Barak, Hebrew University Jerusalem (HUJI)
Hermann Härtig, TU Dresden, Operating Systems Group (TUDOS)
Wolfgang Nagel, TU Dresden, Center for Information Services and HPC (ZIH)
Alexander Reinefeld, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB)

CARSTEN WEINHOLD, TU DRESDEN
The ideal world assumption:

- Identical, predictable, and reliable nodes
- Fast and reliable interconnect
- Balanced applications
- Isolated partitions of fixed size
Fixed-size chunks of work
One thread per core
Systems software:
optimize communication latency
- Message passing uses polling
- Batch scheduler for start / stop
- Separate servers for I/O
- Small OS on each node

No OS on critical path
Application: CP2K

Computation–communication ratio of CP2K on 512 cores
Application: COSMO-SPECS+FD4

Computation–communication ratio of COSMO-SPECS+FD4 on 128 cores
Application: COSMO-SPECS+FD4

Unbalanced compute times of ranks per time step

Hand balanced compute times of ranks per time step

Computation–communication ratio of COSMO-SPECS+FD4 on 128 cores
Application: PRIME

Unbalanced compute times of ranks per time step

Now think of:
- Composite applications
- In-situ visualization, etc.
FFMK: A Fast and Fault-Tolerant Microkernel-Based Operating System for Exascale Computing

Deutsche Forschungsgemeinschaft

German Priority Programme 1648 Software for Exascale Computing
- Dynamic applications & platforms
- Increased fault rates
- Power / Dark silicon
- Heterogeneity (cores, memory, ... )
A Microkernel-Based OS for Exascale Computing
3 ABSTRACTIONS

Light-weight Kernel (L4)
MESSAGE PASSING

A Microkernel-Based OS for Exascale Computing

Light-weight Kernel (L4)

Device Interrupt
**Wake from interrupt on L4:** 900 cycles, **0.3 µs**

(best case, on Intel Core i7-4770 CPU @ 3.40GHz)
A Microkernel-Based OS for Exascale Computing
Real-time

Security: small Trusted Computing Base

Resilience: small Reliable Computing Base
HYBRID SYSTEM

Light-weight Kernel (L4)

File
Network
I/O

Service OS

uncritical complex

critical simple
A Microkernel-Based OS for Exascale Computing
The Hebrew University
A Microkernel-Based OS for Exascale Computing

**Driver Reuse**

- **Light-weight Kernel (L4)**
  - L4 App
    - `libibverbs`
    - User-space Driver
    - I/O
  - Proxy App
    - Msg Buffer 1
    - Msg Buffer 2
    - Proxy App
    - I/O

- **Linux Kernel**
  - `/dev/ib0`
  - IB Core
  - Kernel Driver

- **I/O**
A Microkernel-Based OS for Exascale Computing

NODE ARCHITECTURE

Light-weight Kernel (L4)

MPI Library

Infiniband

Runtime

Checkpoint

MPI Proxies

Infiniband

Linux Kernel

Compute cores

Service cores

Application

Service OS

Application

Application
### MTBF for Component Failures in an HPC System

<table>
<thead>
<tr>
<th>Failed Component</th>
<th># of Nodes Affected</th>
<th>MTBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFS, core switch</td>
<td>1408</td>
<td>65.10 days</td>
</tr>
<tr>
<td>Rack</td>
<td>32</td>
<td>86.90 days</td>
</tr>
<tr>
<td>Edge switch</td>
<td>16</td>
<td>17.37 days</td>
</tr>
<tr>
<td>PSU</td>
<td>4</td>
<td>28.94 days</td>
</tr>
<tr>
<td>Compute nodes</td>
<td>1</td>
<td>15.8 hours</td>
</tr>
</tbody>
</table>

Kento Sato et al., „Design and Modeling of a Non-blocking Checkpointing System“, SC’12
- **In-memory XtreemFS volume** for application-level checkpointing
- **RAID-5 erasure coding**: recovery with 1 failed OSD
- Demonstrator running BQCD code on a Cray XC30
Raghunath Rajachandrasekar et al., A 1 PB/s File System to Checkpoint Three Million MPI Tasks, HPDC’13
OVERDECOMPOSITION

processing units

Barrier
OVERDECOMPOSITION

Barrier

processing units

time
OVERDECOMPOSITION

processing units

time

Barrier
OVERDECOMPOSITION

COSMO-SPECS+FD4 (unbalanced, HT)

Oversubscribed runs: 32–256 MPI ranks on 4 quad-core nodes (w/ polling)
Busy waiting = Computation
COSMO-SPECS+FD4 (unbalanced, HT)

Polling (busy waiting)

Oversubscribed runs: 32–256 MPI ranks on 4 quad-core nodes (w/ polling)
OVERDECOMPOSITION

- COSMO-SPECS+FD4 (balanced, no HT)
- COSMO-SPECS+FD4 (unbalanced, no HT)

Oversubscribed runs: 16–256 MPI ranks on 4 quad-core nodes (w/o polling)
CP2K (unbalanced, no HT)

Oversubscribed runs: 16–256 MPI ranks on 4 quad-core nodes (w/o polling)
OVERDECOMPOSITION

With MPI:
- Do not: busy wait (except very shortly)
- Do: Block in kernel
- Needs: fast unblocking of threads, when message comes in
- We build: shortcut from IB driver into MPI threads (no Linux!)
CENTRALIZED
A Microkernel-Based OS for Exascale Computing
GOSSIP SCALABILITY

Low overhead:
No noticeable overhead at gossip interval of 64-256 ms

Quality of information:
Average age at nodes in the order of 2-3 s with gossip interval of 256 ms


A Microkernel-Based OS for Exascale Computing
Table 6: Simulations of the average master’s age with failed nodes.

<table>
<thead>
<tr>
<th>Number of circulating local failed nodes per colony</th>
<th>Circulating \textit{local windows} of size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>0</td>
<td>11.74</td>
</tr>
<tr>
<td>1</td>
<td>11.71</td>
</tr>
<tr>
<td>2</td>
<td>11.75</td>
</tr>
<tr>
<td>4</td>
<td>11.81</td>
</tr>
<tr>
<td>8</td>
<td>11.83</td>
</tr>
<tr>
<td>16</td>
<td>11.95</td>
</tr>
<tr>
<td>32</td>
<td>12.12</td>
</tr>
</tbody>
</table>

| Standard deviation | 0.49 | 0.42 | 0.37 | 0.36 | 0.36 |
| Increase rate      | 3.2%  | 3.9%  | 3.5%  | 3.4%  | 3.6%  |

Average age at master (1024 nodes per colony)

**Gossip is fault tolerant:**

Only slight increase in average age when substantial number of nodes fail (up to 32 of 1024 in each colony)

A Microkernel-Based OS for Exascale Computing
When

**MOSIX:** Load difference discovered
  + Anomaly anticipated

Where

**MOSIX:** Memory, cycles, comm
  + Topology

Which

**MOSIX:** Past predicts future
  + Application knowledge
A Microkernel-Based OS for Exascale Computing
MANAGEMENT

Resource Prediction

Decision Making

Migration

Hardware monitoring

Platform info (Gossip)

Past behavior

Application hints

Future behavior?
Resource prediction:
- CPU cycles
- Cache misses
- Memory
- Energy?
THERMAL PLANNING

Barrier

processing units

time
THERMAL PLANNING

Barrier

processing units

time
THERMAL PLANNING

Barrier

processing units

time
Resource prediction:
- CPU cycles
- Cache misses
- Memory
- Energy?

Fault tolerance:
- Which data to checkpoint (and when)
- Ability to handle node failures
A Microkernel-Based OS for Exascale Computing
FFMK prototype runs on real HPC system

We build **OS/R for Exascale:**
- Load management
- Fault tolerance
- Take burden from application developers

ffmk.tudos.org