Systems@ETH zürich

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Database-Operating System Co-Design

Jana Giceva

supervised by
Prof. Dr. Gustavo Alonso

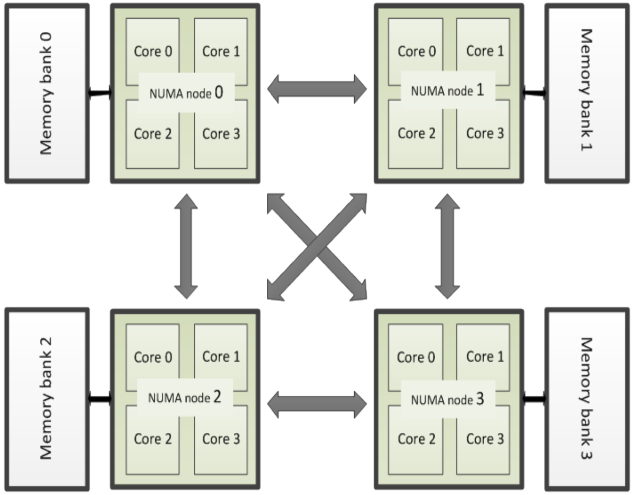
Motivation

- Diversity in hardware resources
  - Scalability
  - Heterogeneity
- Factors influencing database research
  - New hardware trends
  - Diverse workloads
  - Application requirements and constraints
- Database appliances
  - Specifically tailored hardware, operating system and underlying software
  - Cross-layer optimizations

Heterogeneity in modern hardware: NUMA

- NUMA definition
  - Non-Uniform Memory Architecture

- AMD Shanghai NUMA layout



- NUMA awareness and its impact on performance:
  - Aware and unaware memory access and DRAM utilization
  - NUMA effects on performance

Problem statement

- Building blocks:
  - Barrelfish OS [1]
    - Addresses hardware trends: multicore scalability and heterogeneity
  - CSCS engine [2]
    - Addresses workload demands and application requirements
- Questions we aimed to answer:
  - Porting the CSCS engine on Barrelfish
    - Challenges?, Modifications?
  - Performance
    - Detailed analysis? Varying factors? Hot-spots? How does it scale?
  - Nature and characteristics
    - Compared to baseline run on Linux?
  - How does it perform on other architectures?
  - Ideas for future work?

[1]Barrelfish: http://barrelfish.org/ [2] CSCS engine: Shared Scans on Column Stores: contact Tudor Salomie

The CSCS engine

- Column stores:
  - Serialize column values together
  - Engines using column stores:
    - Vertica, C-Store, MonetDB, SAP T-Rex
- Shared scans:
  - Multi-query optimizations[1]
  - Shared scans:
    - RedBrick, IBM Blink, Crescando
  - Crescando's ClockScan
- Shared scans on column stores
  - Main-memory CSCS engine

[1] Timos K.Sellis. Multiple-query optimization. ACM Trans. Database Syst., 13:23-52, March 1988

The Barrelfish OS

- Implementation of a Multikernel:
  - Treats machines as a network of cores
  - Explicit message passing communication
  - No sharing rather replication and partitioning
- Handling hardware heterogeneity:
  - Hardware transparency for the applications
  - System Knowledge Base (SKB) service
  - Delegate resource allocation to the applications
- Scheduling
  - At multiple timescales: long-, medium- and short-term

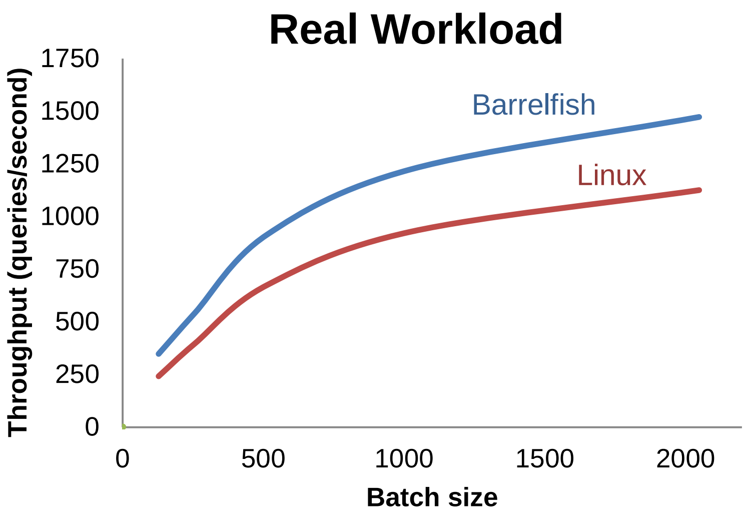
Porting the CSCS engine on Barrelfish

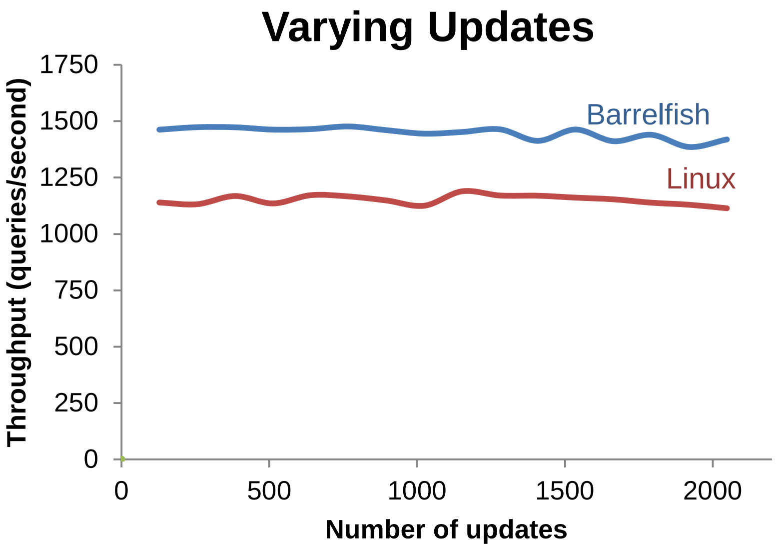
- Differences:
  - Kernel, C library, C++ library
- Dependencies:
  - Boost library, Google hash library
- Challenges:
  - CMake to Hake conversion
  - C++ support: exceptions
  - Threads and synchronization primitives
  - Memory allocation service
  - Networking stack driver implementation

Experiment setup

- Workload:
  - Amadeus - millions of flight bookings, 1 relation, 48 attributes
    - **real workload:** constant ratio of queries and updates in a batch
    - **synthetic workload:** varying updates, read only workload
- Machines:
  - **AMD SantaRosa**: 2x2-core, 2.8GHz Opteron 2220, local memory controller, 2 HyperTransport links, 1MB L2 cache
  - **AMD Shanghai**: 4x4-core, 2.5GHz Opteron 8380, 4 HyperTransport links, 512kB local L2 cache and 6MB shared L3 cache
- Performance measurement:
  - Performance counter events
  - Barrelfish: caliper mode, reading from registers
  - Linux: sampling mode, OProfile

Baseline results

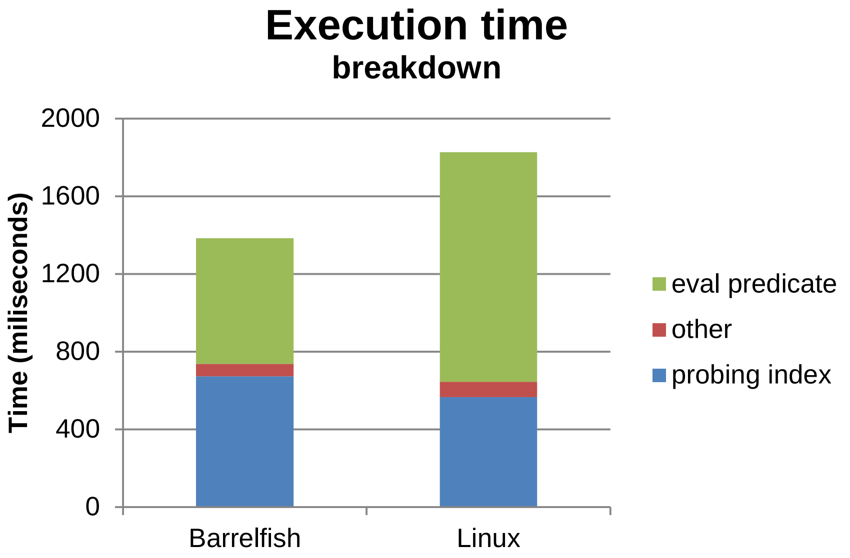




- In both workloads:
  - The throughput curves of Linux and Barrelfish look alike
- Varying the number of updates:
  - Does not impose performance degradation
- Varying the datastore size (not shown):
  - Throughput is directly proportional to the datastore size loaded

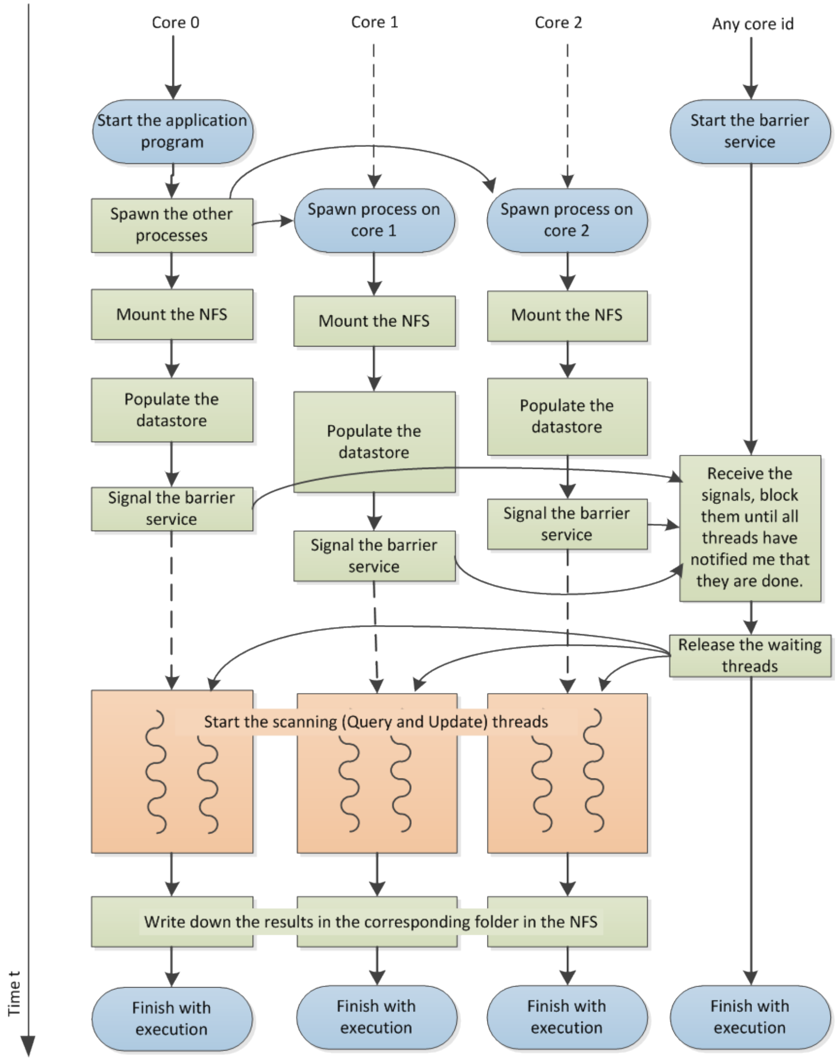
Linux vs. Barrelfish

- Performance analysis:
  - For both Linux and Barrelfish
  - Measured performance values
    - Low L1,L2 and L3 miss rates
    - Low DTLB miss rate
    - Good IPC/CPI values
  - CPU bound
- Execution time breakdown:
  - Evaluating predicate demands string comparison
  - Different C and C++ library implementation
- Bandwidth utilization:
  - Linux vs Barrelfish: ~30% difference
  - Same number of bytes transferred
  - Difference due to execution time length

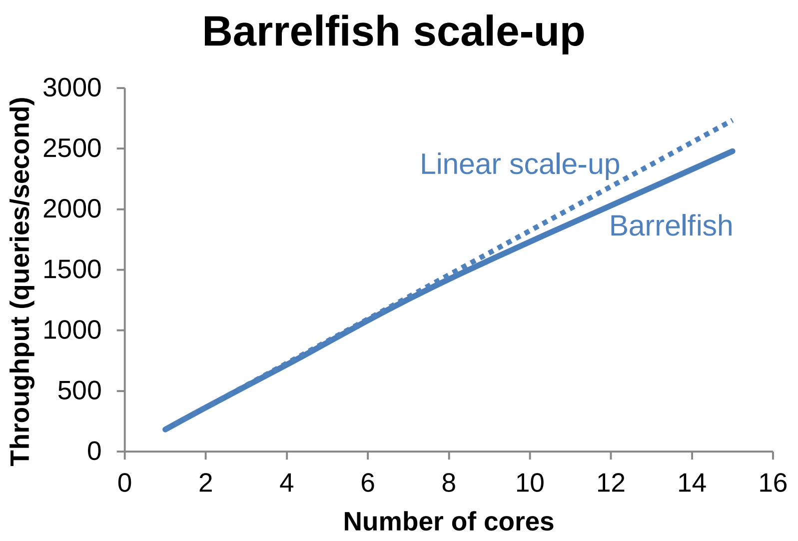


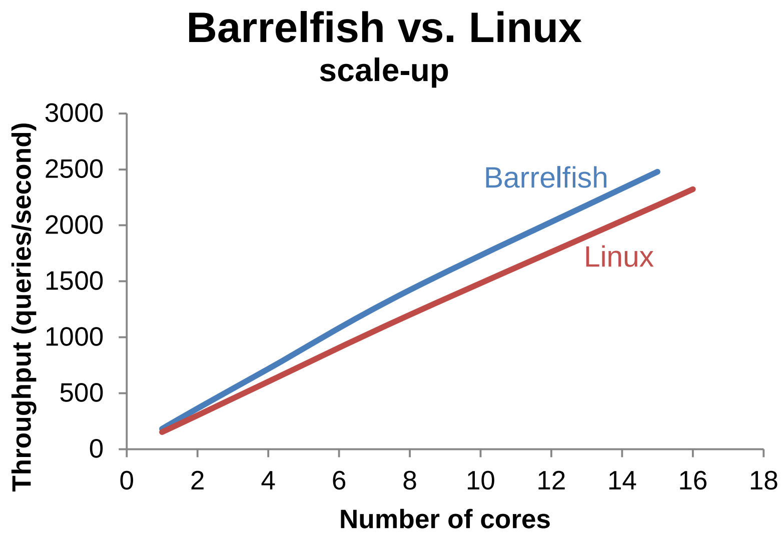
Scale-up implementation

- Multiple designs considered:
  - multithreaded design
  - processes sharing memory region
  - processes mounting the NFS
- Implemented design:
  - Multiple processes mount the NFS
  - … using the new network driver
  - Barrier service synchronizes the runs



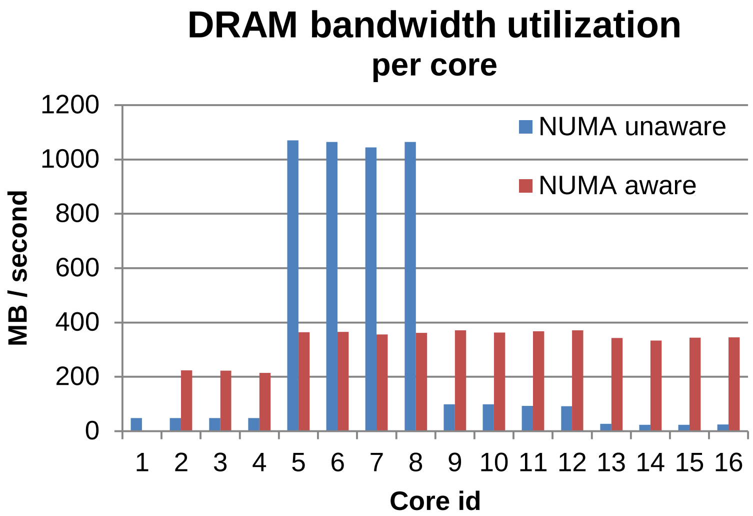
Performance on multiple cores

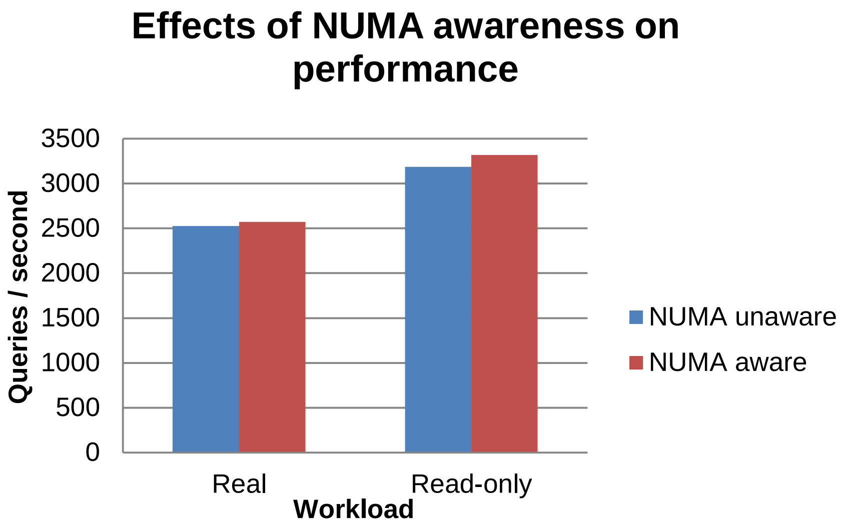




- In both Linux and Barrelfish:
  - CSCS's throughput scales almost linearly with the number of cores

NUMA analysis





- NUMA unaware memory access:
  - Non uniform utilization of DRAM bandwidth and memory controller
  - All pressure on the second NUMA node (cores 5 – 8)
- NUMA awareness:
  - Better distribution of the memory access requests and serves
  - Does not provide significant performance improvement

Thesis contribution

- The CSCS engine works on top of Barrelfish
  - Extensive performance analysis
  - Performance and behavior resembles the baseline of CSCS's run on Linux
  - More details in the thesis report[1] and on the project wiki page[2]

- Solid foundation for future work
  - Can be enhanced with more complex features
  - … that will use the services provided by Barrelfish

  - Eventually resulting in a fully functional database
  - … tightly collaborating with the OS via well defined interfaces

[1] Thesis report: https://trac.systems.inf.ethz.ch/trac/systems/mcdb/attachment/wiki/ClockScanningColumnStoresBarrelfish/Thesis.pdf

[2] Wiki page: https://trac.systems.inf.ethz.ch/trac/systems/mcdb/wiki/ClockScanningColumnStoresBarrelfish

Ideas for future work

- Open questions
  - When are we going to hit the point where scale-up throughput will no longer follow the linear scalability line?
  - What resource bottleneck are we going to hit then?

- Ideas for future work
  - Short run:
    - Test on more different architectures. Especially with more than 30 cores
    - Define interfaces for communicating with the SKB for proper resource allocation and deployment
  - Long run:
    - Extend the CSCS with more complex features that will exercise Barrelfish functionality for scheduling and the SKB service
    - Extend the SKB service to provide more online information