PC² PADERBORN CENTER FOR PARALLEL COMPUTING

Bachelor Thesis: A Fair $\mathcal{O}(1)$ High Throughput CPU-Scheduler for Linux (HTFS)

Markus Pargmann

University of Paderborn

December 12, 2011

Motivation

Server market goals

- High performance
- Low costs
- Efficient power usage

Server market trends

- Virtualization
- Cloud computing
- Linux (no license costs)

Motivation

Server market goals

- High performance
- Low costs
- Efficient power usage

Server market trends

- Virtualization
- Cloud computing
- Linux (no license costs)

Motivation

Linux

- Kernel is important factor for performance
- Every process needs the CPU
- CPU-Scheduler manages most important component
- Completely Fair Scheduler is $\mathcal{O}(\log n)$

Scheduler improvement potential

- Server center with 1000 servers
- 0.1% improvement $= 1$ server
- 1% improvement $= 10$ servers

Motivation

Linux

- Kernel is important factor for performance
- Every process needs the CPU
- CPU-Scheduler manages most important component
- Completely Fair Scheduler is $\mathcal{O}(\log n)$

Scheduler improvement potential

- Server center with 1000 servers
- 0.1% improvement = 1 server
- 1% improvement = 10 servers

PADERBORN CENTER FOR PARALLEL COMPUTING

Outline

Basics

Linux scheduler

- A task is process or thread with priority/weight
- Interactivity: Often switching between states
- Runqueue stores running tasks
- One runqueue per CPU

Completely Fair Scheduler (CFS)

- Virtual runtimes
  - Comparable
  - $vruntime = \frac{runtime}{weight}$
- Tasks sorted by virtual runtimes $\Rightarrow \mathcal{O}(\log n)$
- Choose task with lowest virtual runtime
- Simple design
- High constant time consumption

Basics

Linux scheduler

- A task is process or thread with priority/weight
- Interactivity: Often switching between states
- Runqueue stores running tasks
- One runqueue per CPU

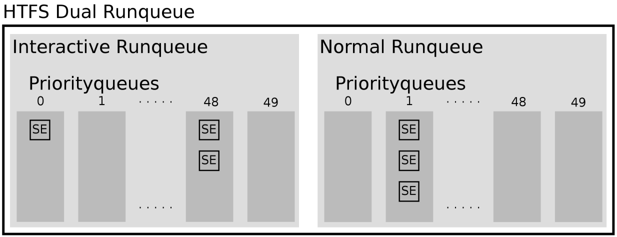
Completely Fair Scheduler (CFS)

- Virtual runtimes
  - Comparable
  - $vruntime = \frac{runtime}{weight}$
- Tasks sorted by virtual runtimes $\Rightarrow \mathcal{O}(\log n)$
- Choose task with lowest virtual runtime
- Simple design
- High constant time consumption

Outline

Concepts – Multi-Queue Design

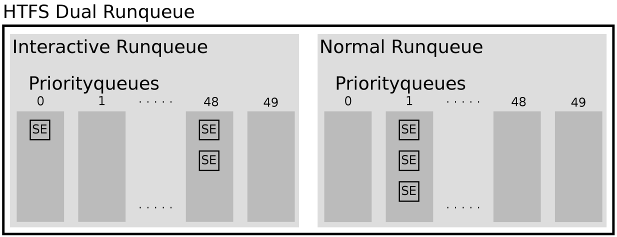
- Interactive/Normal Runqueue
- 50 priorityqueues each
- Priorityqueue is list of entities



- Each priorityqueue / runqueue can be interpreted as one entity
- Using a runcost-balance model to choose
- All basic operations in constant time
- Interactive runqueue has more weight

Concepts – Multi-Queue Design

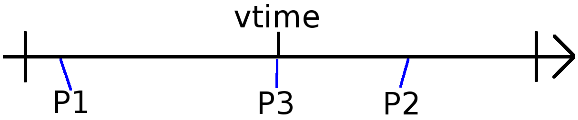
- Interactive/Normal Runqueue
- 50 priorityqueues each
- Priorityqueue is list of entities



- Each priorityqueue / runqueue can be interpreted as one entity
- Using a runcost-balance model to choose
- All basic operations in constant time
- Interactive runqueue has more weight

Concepts – Virtual Time

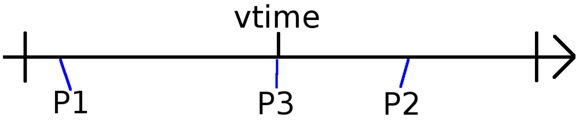
- Virtual time to compare virtual runtimes
- Task's virtual runtime initialized with virtual time
- Virtual runtime only changes after running
- Virtual time changes after running any task



- Information about task's situation through comparison
- Adaptive weight based on virtual runtime
- All virtual runtimes are within boundaries

Concepts – Virtual Time

- Virtual time to compare virtual runtimes
- Task's virtual runtime initialized with virtual time
- Virtual runtime only changes after running
- Virtual time changes after running any task



- Information about task's situation through comparison
- Adaptive weight based on virtual runtime
- All virtual runtimes are within boundaries

Concepts – Fairness Proof Idea

HTFS fairness

- Case vruntime $<<$ vtime
  - Minimal possible vruntime increase: $\frac{\frac{1}{8} \cdot timeslice}{weight}$
  - Maximal possible boost: $>$ 16 (interactive runqueue)
  - $\Rightarrow 16 \cdot \frac{\frac{1}{8} \cdot timeslice}{weight} >$ vtime increasement
- Case vruntime $>>$ vtime
  - Only case: inaccurate time measurement
  - Can be compensated by weight decreasement

CFS fairness

- Executes the task with smallest vruntime
- $\Rightarrow$ always within boundaries

Concepts – Fairness Proof Idea

HTFS fairness

- Case vruntime $<<$ vtime
  - Minimal possible vruntime increase: $\frac{\frac{1}{8} \cdot timeslice}{weight}$
  - Maximal possible boost: > 16 (interactive runqueue)
  - $\Rightarrow 16 \cdot \frac{\frac{1}{8} \cdot timeslice}{weight} >$ vtime increasement
- Case vruntime $>>$ vtime
  - Only case: inaccurate time measurement
  - Can be compensated by weight decreasement

CFS fairness

- Executes the task with smallest vruntime
- $\Rightarrow$ always within boundaries

Concepts – Fairness Proof Idea

HTFS fairness

- Case vruntime $<<$ vtime
  - Minimal possible vruntime increase: $\frac{\frac{1}{8} \cdot timeslice}{weight}$
  - Maximal possible boost: > 16 (interactive runqueue)
  - $\Rightarrow 16 \cdot \frac{\frac{1}{8} \cdot timeslice}{weight} >$ vtime increasement
- Case vruntime $>>$ vtime
  - Only case: inaccurate time measurement
  - Can be compensated by weight decreasement

CFS fairness

- Executes the task with smallest vruntime
- $\Rightarrow$ always within boundaries

PADERBORN CENTER FOR PARALLEL COMPUTING

Concepts – Fairness Proof Idea

HTFS fairness

- Case vruntime $<<$ vtime
  - Minimal possible vruntime increase: $\frac{\frac{1}{8} \cdot timeslice}{weight}$
  - Maximal possible boost: >16 (interactive runqueue)
  - $\Rightarrow 16 \cdot \frac{\frac{1}{8} \cdot timeslice}{weight} >$ vtime increasement
- Case vruntime $>>$ vtime
  - Only case: inaccurate time measurement
  - Can be compensated by weight decreasement

CFS fairness

- Executes the task with smallest vruntime
- $\Rightarrow$ always within boundaries

Outline

Evaluation – Test Environment

Evaluation goals

- Measure throughput and interactivity
- Use different workloads
- Use different hardware platforms

Environment

- Phoronix test suite
  - Many benchmarks
  - Records scheduler statistics
- Athlon II M320 Dual-Core @ 2.1GHz
- 2 x Intel Xeon E5520 @ 2.26GHz

Evaluation – Test Environment

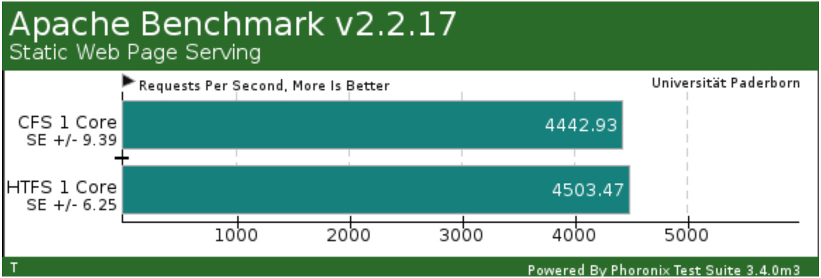
Evaluation goals

- Measure throughput and interactivity
- Use different workloads
- Use different hardware platforms

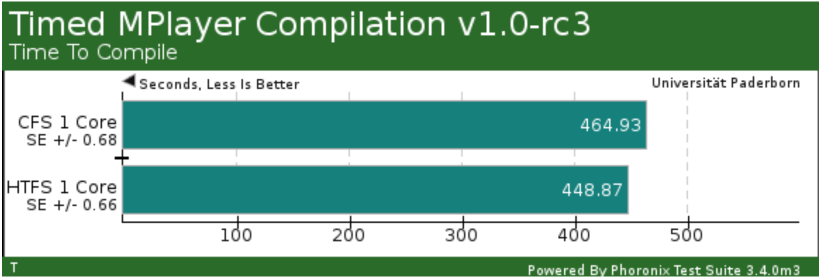
Environment

- Phoronix test suite
    - Many benchmarks
    - Records scheduler statistics
- Athlon II M320 Dual-Core @ 2.1GHz
- 2 x Intel Xeon E5520 @ 2.26GHz

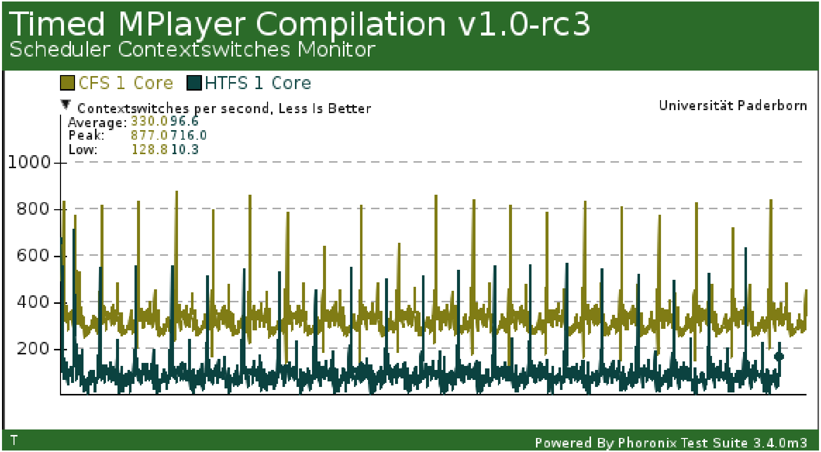
Evaluation – Performance



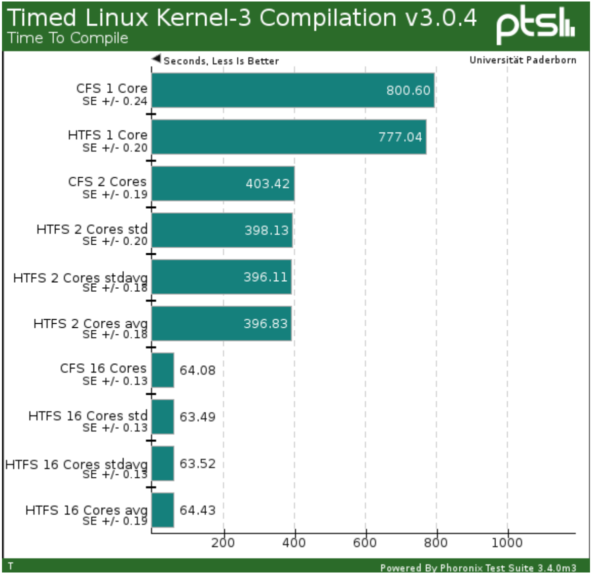
Evaluation – Performance



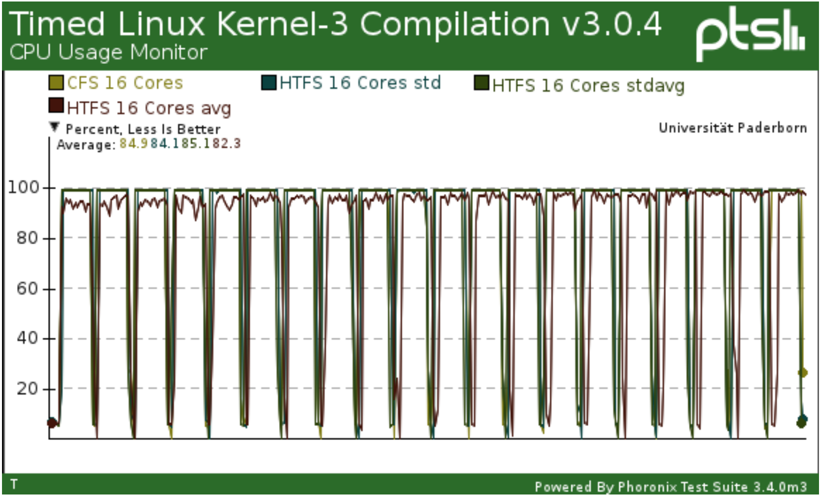
Evaluation – Performance



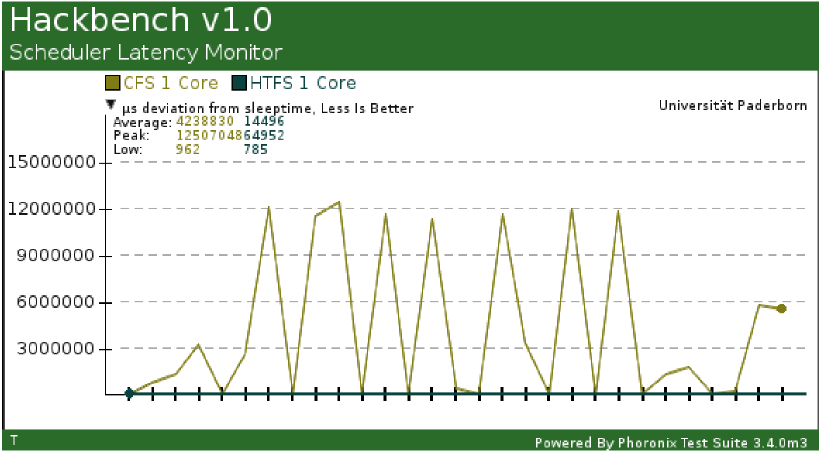
Evaluation – Multi-Core Performance



Evaluation – Multi-Core Performance



Evaluation – Interactivity



Evaluation – Summary

Single Core	HTFS
CFS better than . . .	31.7%
. . . better than CFS	51.7%

Dual Core	HTFS		
	std	stdavg	avg
CFS better than . . .	41.7%	20.0%	40.0%
. . . better than CFS	50.0%	66.7%	47.5%

16 Core	HTFS		
	std	stdavg	avg
CFS better than . . .	36.7%	42.2%	44.4%
. . . better than CFS	63.3%	55.6%	50.0%

Outline

Conclusion

Pros

- Low constant time consumption
- $\mathcal{O}(1)$ time complexity
- Good interactivity in all tests
- Good single core performance

Cons

- Flaws with current multi core performance of HTFS and CFS
- Possibly interactivity problems in some special cases

Conclusion

Pros

- Low constant time consumption
- $\mathcal{O}(1)$ time complexity
- Good interactivity in all tests
- Good single core performance

Cons

- Flaws with current multi core performance of HTFS and CFS
- Possibly interactivity problems in some special cases

Outlook

- The design is a good alternative
- Especially servers could improve efficiency

Future work

- Some code improvements (Load balancing)
- New model for taskgroup support
- New preemption handling

Questions?

Thank you for listening, questions?

Questions?

Concepts – Timeslices

HTFS

- Global interactive timeslice of $2^{17}ns = 131\mu s$
- CPU-wide adaptive timeslice of $2^x, 17 \leq x \leq 30$
- x is decreased if interactive task's latency too high
- x is increased if latencies okay

Completely Fair Scheduler

- No fixed timeslice
- Process runtime dependent of
  - Virtual runtime of the task
  - Number of tasks
  - Minimum granularity