Karlsruhe Institute of Technology

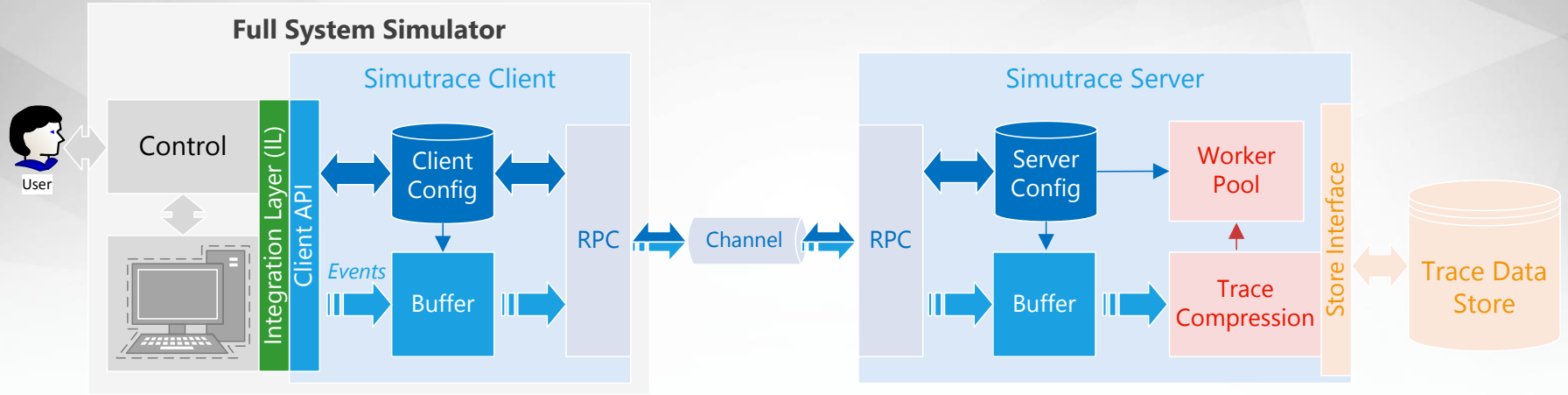
Simutrace
http://simutrace.org/

Efficient Full System Memory Tracing with Simutrace

GI Fachgruppentreffen Betriebssysteme (BS) 2014

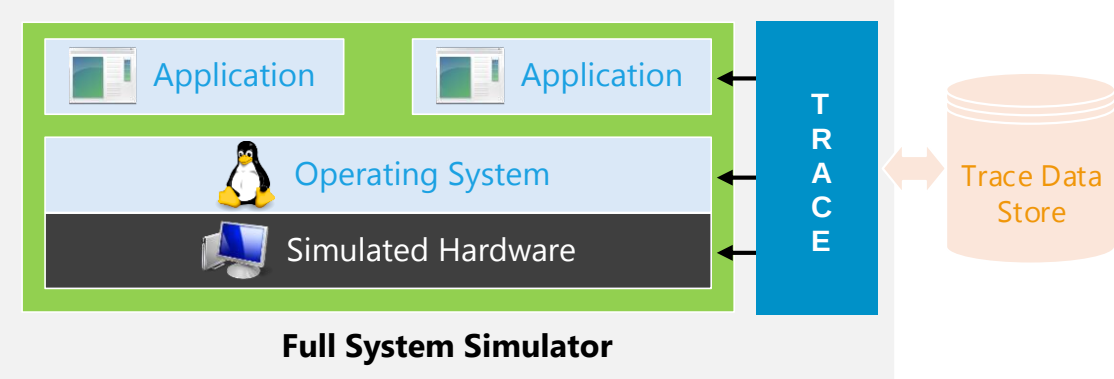
Thorsten Gröninger, Marc Rittinghaus, Frank Bellosa

SYSTEM ARCHITECTURE GROUP
DEPARTMENT OF COMPUTER SCIENCE



Motivation

- Operating system performance analysis
  - Application and kernel interaction
  - Memory access patterns
  - Cache efficiency



- Want to trace and do offline analysis
  - Full system simulation is slow (100x)
  - Repeatability/reproducibility of experiments/results

Memory Tracing

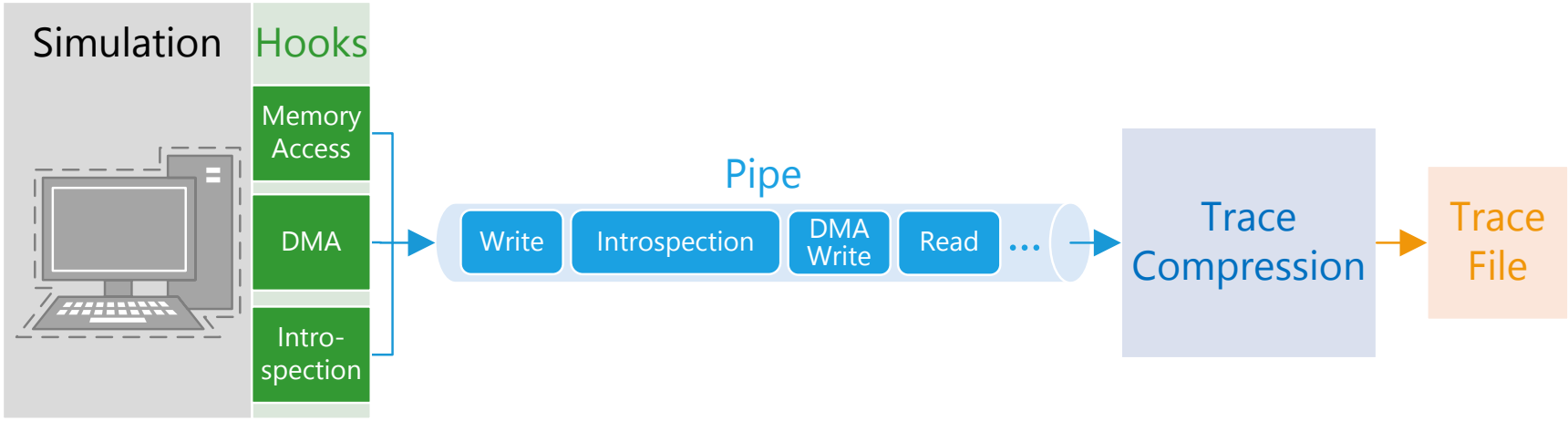
- Want: Record memory accesses and introspection data
  - Correlate operations with processes, memory areas, etc.

- Challenges
  - High event rate: approx. 150 MiB/s (QEmu single-threaded)
  - High amount of data:

Kernel Build	SPEC.h264 (w)	SPEC.gcc
3.6 TiB	11 TiB	8 TiB
150 bil. Entries	360 bil. Entries	310 bil. Entries

We need to compress traces

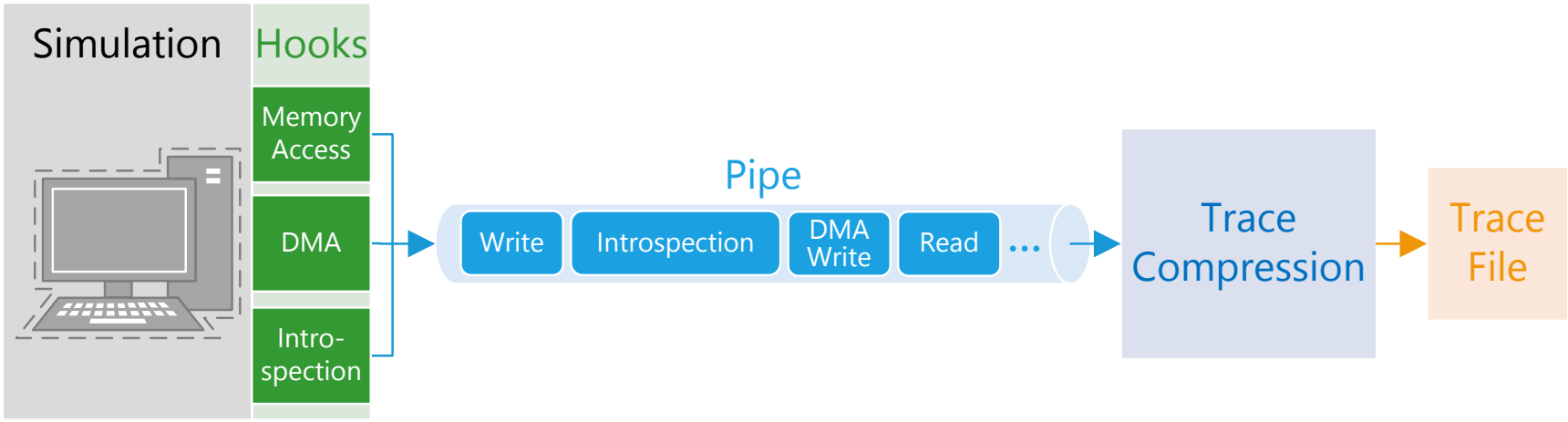
Memory Tracing



- Simple tracer: pipe data into compressor
  - ± Kernel build (3.6 TiB): 700 GiB (bzip2), 800 GiB (deflate)
  - – Simulation slowdown: 7x (deflate)

We need a more sophisticated approach

Simutrace Design Goals



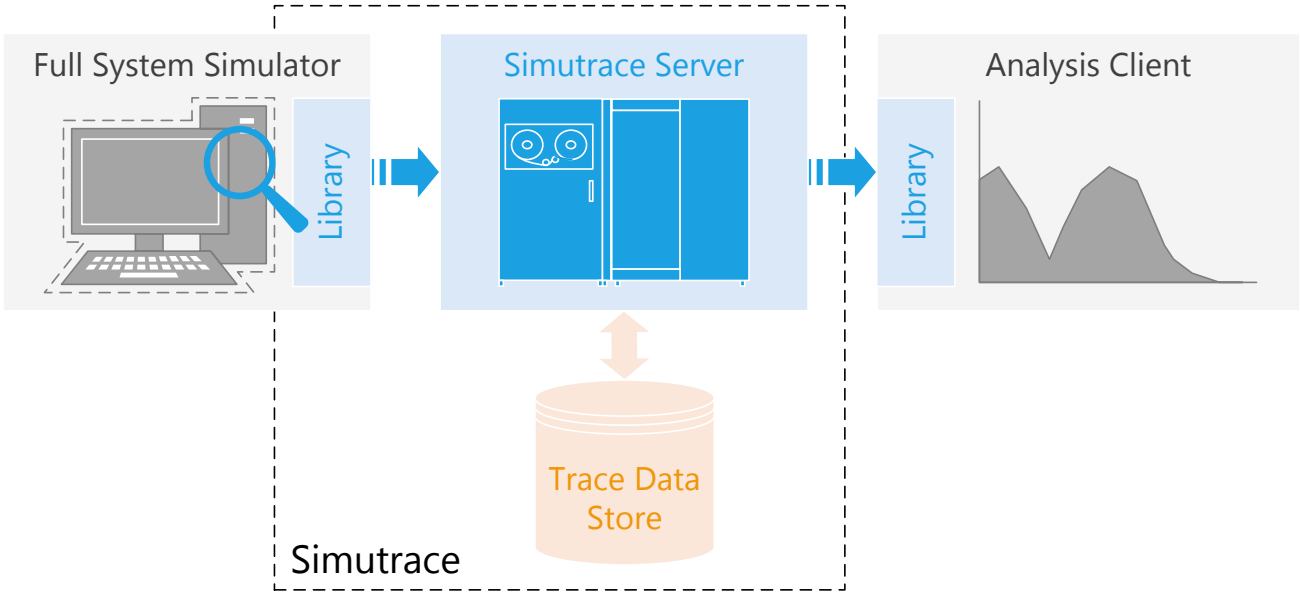
Speed
Keep simulation slowdown low

Flexibility
Trace data of various types

Compression
Leverage special purpose compression if possible

Storage
Seekable storage format

Simutrace Architecture



- Client-server architecture
  - Clients submit or query data
  - Server processes traces and manages storage
- Library in client manages connection
  - Trace data exchange over shared memory or sockets

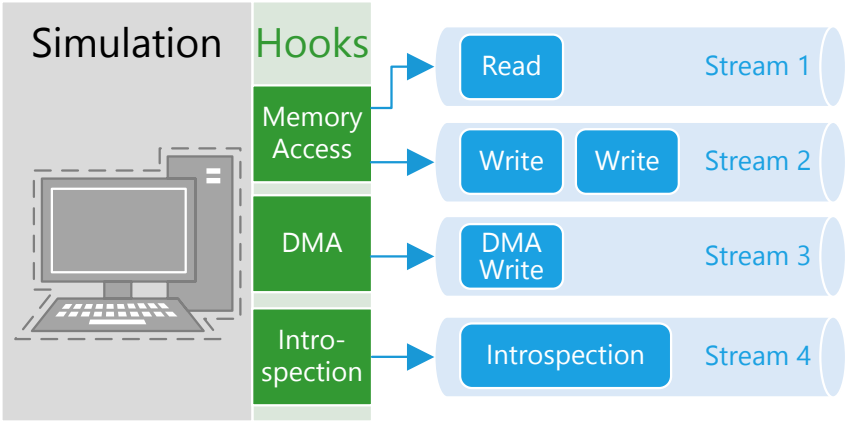
Simutrace – Flexibility

Want: Trace data of various types

- Challenges
  - Entries of different size and compressibility are interleaved
  - Varying number of entries per type

➢ **Separate entries of different type**

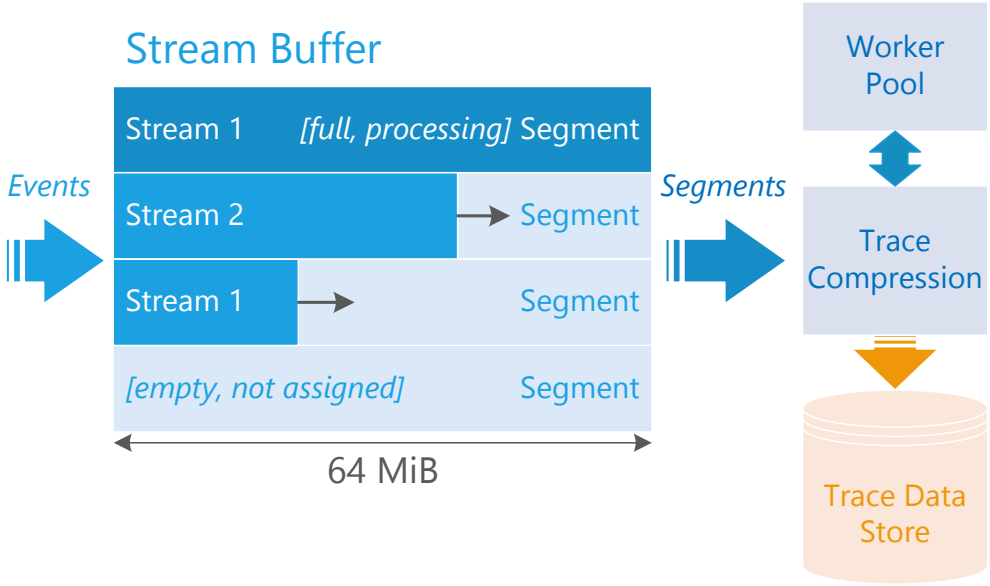
- Streams
  - Contain only *one* type
  - Group semantically connected entries
  - Ease type specific compression
  - Ease addition of further data



Simutrace – Speed

Want: Keep simulation slowdown low

- Asynchronous compression
  - Shared buffer (zero-copy)
  - Segment granularity
- Parallel compression
  - Scales with submission rate



- ➢ Simulation runs at near full speed
  - Simulation slowdown: 1.1x

Simutrace – Compression

Want: Leverage special purpose compression schemes

- Compression scheme depends on stream‘s data type
  - General purpose as default (LZMA)
- Memory access: Modified version of VPC4 (SVPC)
  - Leverages memory access locality
  - Combines prediction-based compression with LZMA
  - Improves compression ratio *and* speed
- ➢ Detailed traces become manageable

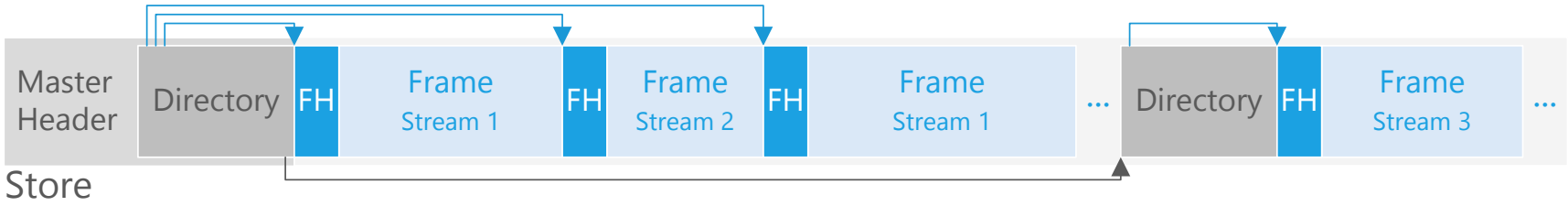
	Kernel Build	SPEC.gcc	memtest
RAW	3.6 TiB	8 TiB	308 GiB
Simutrace	110 GiB (1:32)	70 GiB (1:114)	98 MiB (1:3142)

Simutrace – Storage

Want: Seekable storage format

- Record metadata in trace
  - Simulation time, wall clock time, etc.
  - Index to find position in file

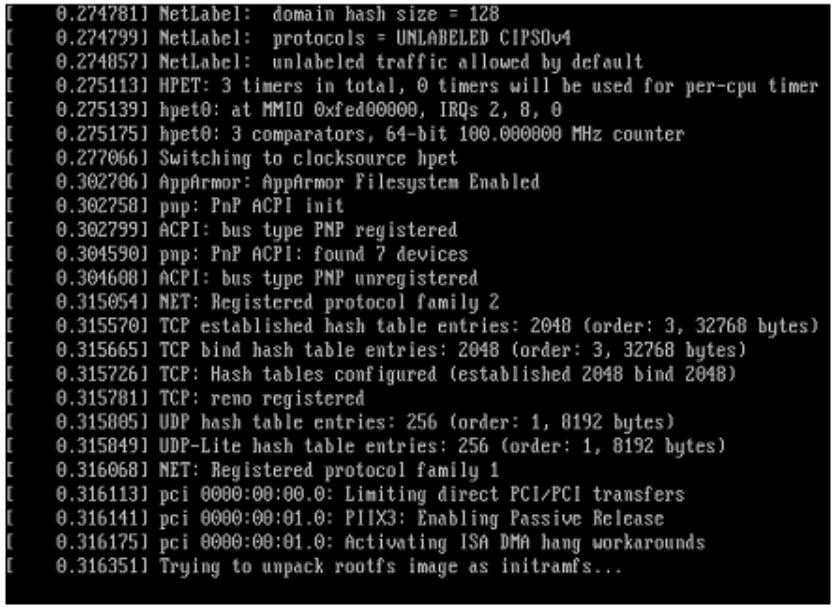
We need partial decompression



- Trace: List of compressed segments
  - \+ Partial decompression
  - \- Segments have varying size
  - \- Segments may be in wrong order

➢ Directory for fast discovery

FrameBuffer Content:

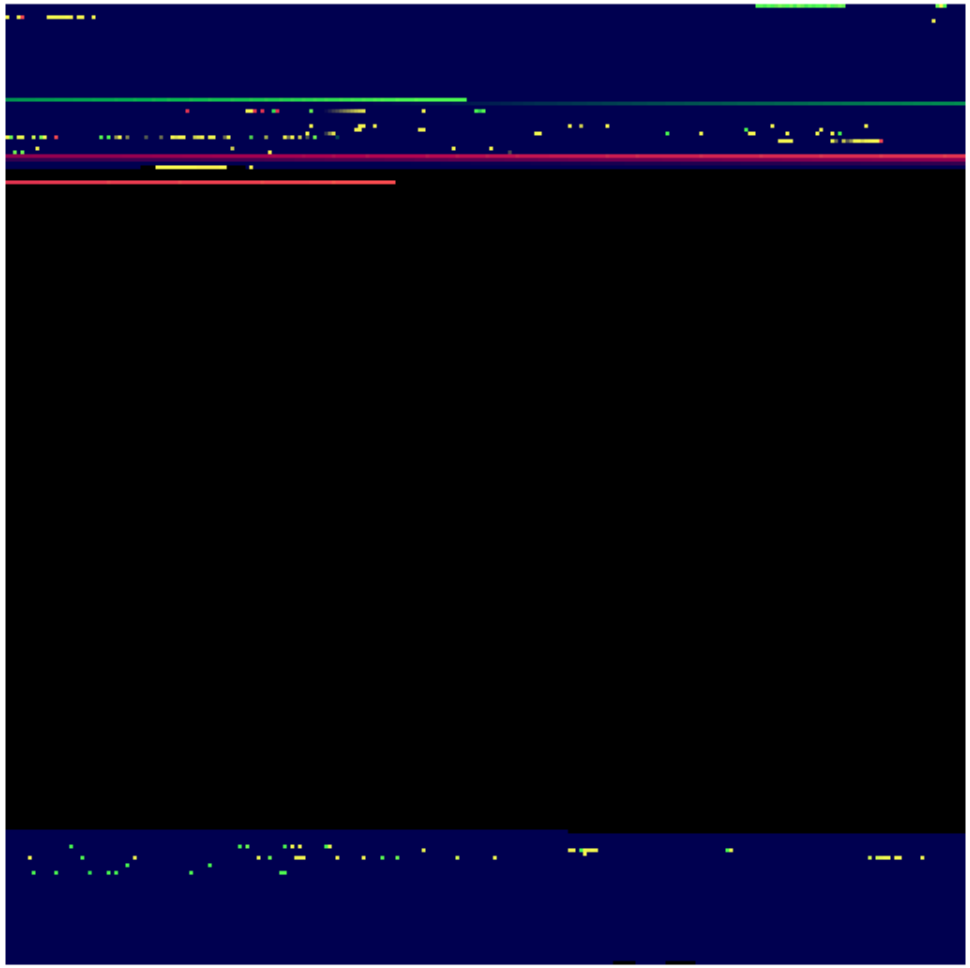


Log:

fork(): swapper/0 (PID: 2) => 5 (kthreadd)
fork(): swapper/0 (PID: 2) => 6 (kthreadd)
fork(): swapper/0 (PID: 2) => 7 (kthreadd)
fork(): swapper/0 (PID: 2) => 8 (kthreadd)
fork(): swapper/0 (PID: 2) => 9 (kthreadd)
fork(): swapper/0 (PID: 2) => 10 (kthreadd)
fork(): swapper/0 (PID: 2) => 11 (kthreadd)
fork(): swapper/0 (PID: 2) => 12 (kthreadd)
fork(): swapper/0 (PID: 2) => 13 (kthreadd)
fork(): swapper/0 (PID: 2) => 14 (kthreadd)
fork(): swapper/0 (PID: 2) => 15 (kthreadd)
fork(): swapper/0 (PID: 2) => 16 (kthreadd)
fork(): swapper/0 (PID: 2) => 17 (kthreadd)
fork(): swapper/0 (PID: 2) => 18 (kthreadd)
fork(): swapper/0 (PID: 2) => 19 (kthreadd)
fork(): swapper/0 (PID: 2) => 20 (kthreadd)
fork(): swapper/0 (PID: 2) => 21 (kthreadd)
fork(): swapper/0 (PID: 2) => 22 (kthreadd)

Timeline:

Physical Memory Content:



CycleCount:	1,582,727,143	Reads:	402,192,802
RealTime:	00:01:18.036		
SimulationTime:	00:00:03.141	Written Data:	1,762,268,455 B
		Processed Data:	20,343,675,216 B
Writes:	326,796,211		
1 Byte Writes:	70,790,401 (21%)	Total Write Entries:	27,858,951,214
2 Byte Writes:	13,868,141 (4%)	Total Read Entries:	47,602,879,275
4 Byte Writes:	68,339,895 (20%)		
8 Byte Writes:	173,797,774 (53%)		

Simutrace

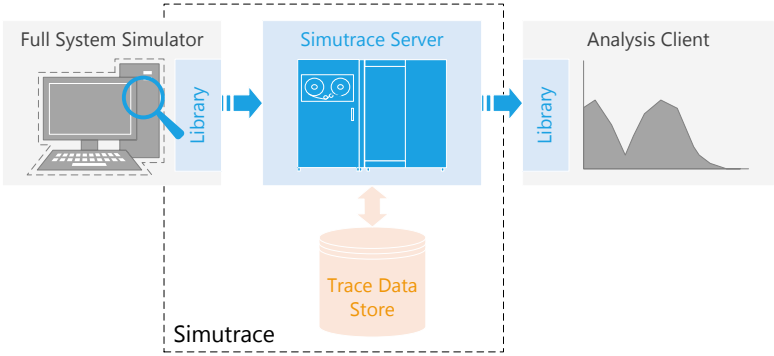
Simutrace

Karlsruhe Institute of Technology

Demo

Conclusion

- Memory tracing puts high pressure on tracing infrastructure
  - Many terabytes of data
  - Billions of entries
- Simutrace: Flexible full system tracing
  - Keeps slowdown at a minimum
  - Delivers high compression
  - Eases access to recorded data

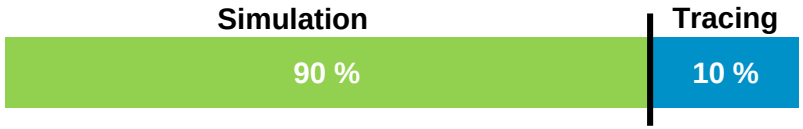


- Will be available as open source
  - http://simutrace.org

Simutrace makes memory tracing efficient

Simutrace in Research

- Student projects at KIT
  - Characteristics of memory duplication
  - Applicability of more accurate page access information
- Analysis of memory duplication on NUMA systems
- SimuBoost



  - Accelerate full system simulation through massive parallelization
  - Significantly increases requirements for tracing infrastructure
    - > 50 simulations in parallel