TAUdb: PerfDMF Refactored

Kevin Huck, Suzanne Millstein,
Allen D. Malony and Sameer Shende

Department of Computer and Information Science
University of Oregon

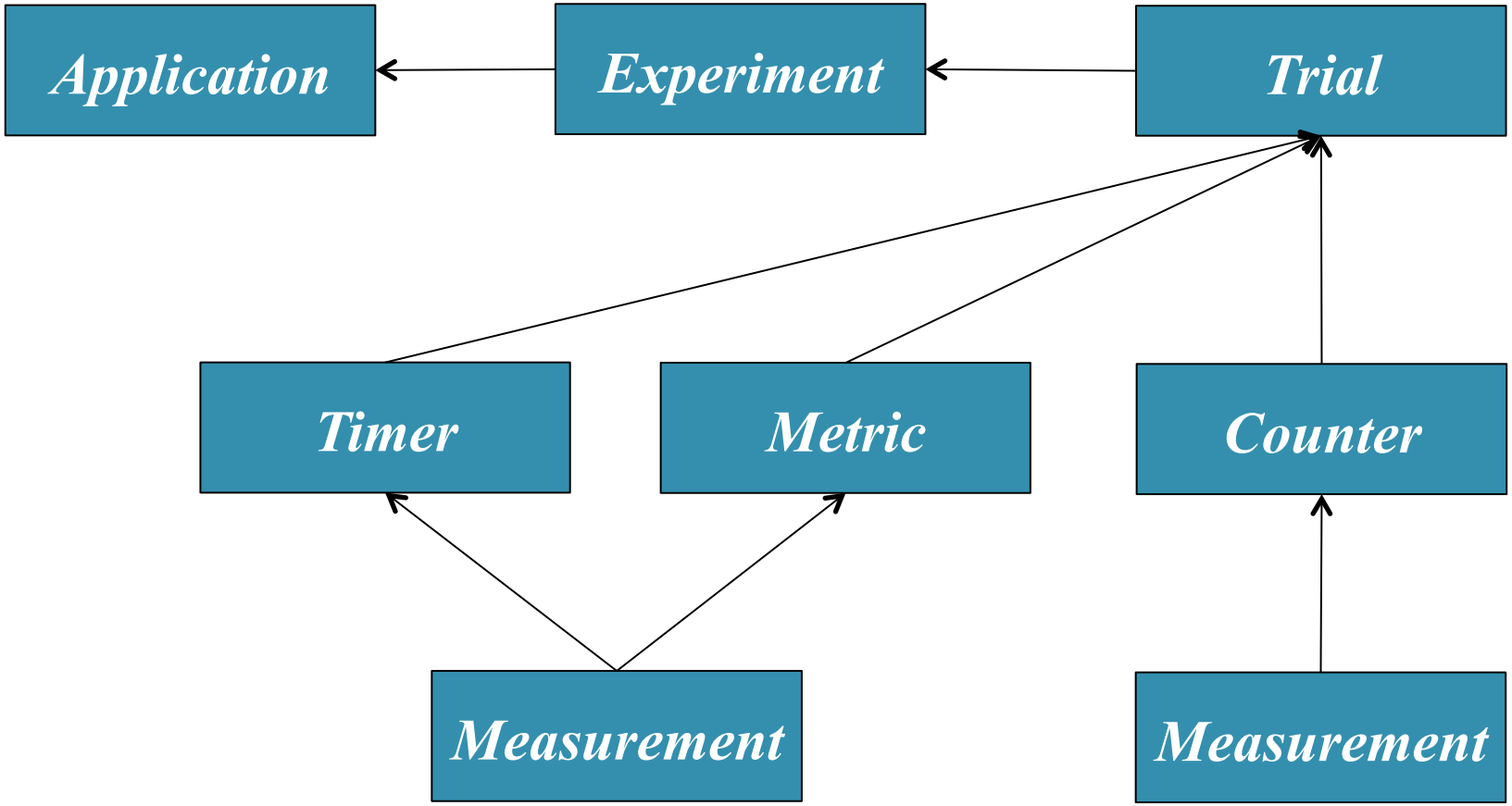
PerfDMF Overview

- Performance Data Management Framework
- Started in 2004 (Huck et al., ICPP 2005)
- Database Schema & Java API (profile parsing, database queries, conversion utilities)
- Provides DB support for TAU Profile Analysis Tools ParaProf, PerfExplorer, EclipsePTP
- Used as regression testing database for TAU
- Used as performance regression database for FACETS (2008-2012)
- Ported to several DBMS: PostgreSQL, MySQL, H2, Derby, Oracle, DB2

Supported Profile Formats

- **TAU** – profiles, packed profiles, snapshots (UO)
- **DynaProf** – PAPI DynaProf profiles (UTK)
- **mpiP** – Lightweight, scalable MPI Profiling (Vetter, Chambreau)
- **HPM Toolkit** profiles (IBM)
- **Gprof** profiles (GNU)
- **PerfSuite** psrun profiles (NCSA)
- **Cube**, **Cube3, Cube4** profiles (FZJ)
- **HPC Toolkit** profiles (Rice)
- **OMPP** – OpenMP Profiler profiles (Fuerlinger)
- **PERI-XML** (PERI)
- **GPTL** – General Purpose Timing Library profiles (ORNL)
- **Paraver** profiles (BSC)
- **IPM** – Integrated Performance Monitoring (NERSC)
- **Google** profiles (Google)
- Others (Gyro, GAMESS, other application-specific timer data)

PerfDMF Schema Overview



Nice and simple, but there are problems...

A Little CScADS History…

- Meeting: "Performance Tools for Petascale Computing" July 21-24, 2008
- PERI-XML Working Group: Towards a Common Exchange Format
- Main Focus: Each data point has a connection to Five profile dimensions
  - Code (static location, binary/source) – Check!
  - Space (physical and logical) – Check!
  - Metrics (data which is collected, derived values) – Check!
  - Dynamic State (callstack, context, …) – um, not explicitly
  - Time (timeline) – no

PerfDMF ~~Complaints~~ ~~Problems~~ "Challenges"

- Metadata is not a first-class citizen in schema
  - Compressed XML document, not context-sensitive
- Hierarchy inadequate (too many/few levels)
- Not enough ***explicit*** semantic relationships in the data
  - Callpaths, phases, timer groups, etc.
- No ***explicit*** support for "special cases"
  - Callpaths, phases, parameters*, snapshots/timestamps
- Inefficiencies
  - Space (some poor normalization)
  - Time (parsing XML slow, loading big trials slow, etc)
- No C API

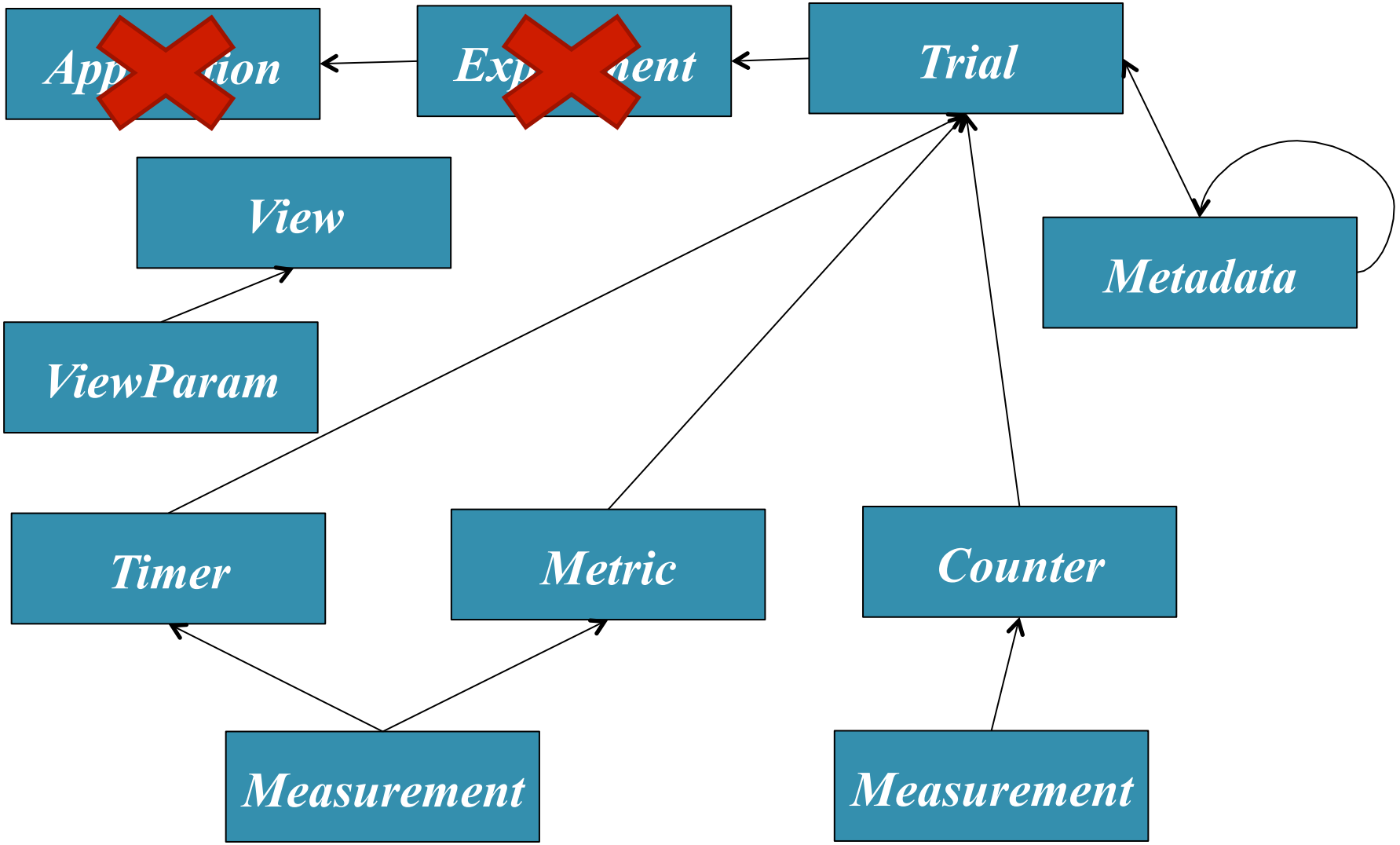
TAUdb – PerfDMF, but better!

- New Schema – (hopefully) all problems addressed
- TAU Tools retain compatibility with old schema
- Refactoring Java API
- Redesigned TAU measurement interface
- Full-featured C API
- SQLite support (evaluating)

TAUdb Schema Organizational Change

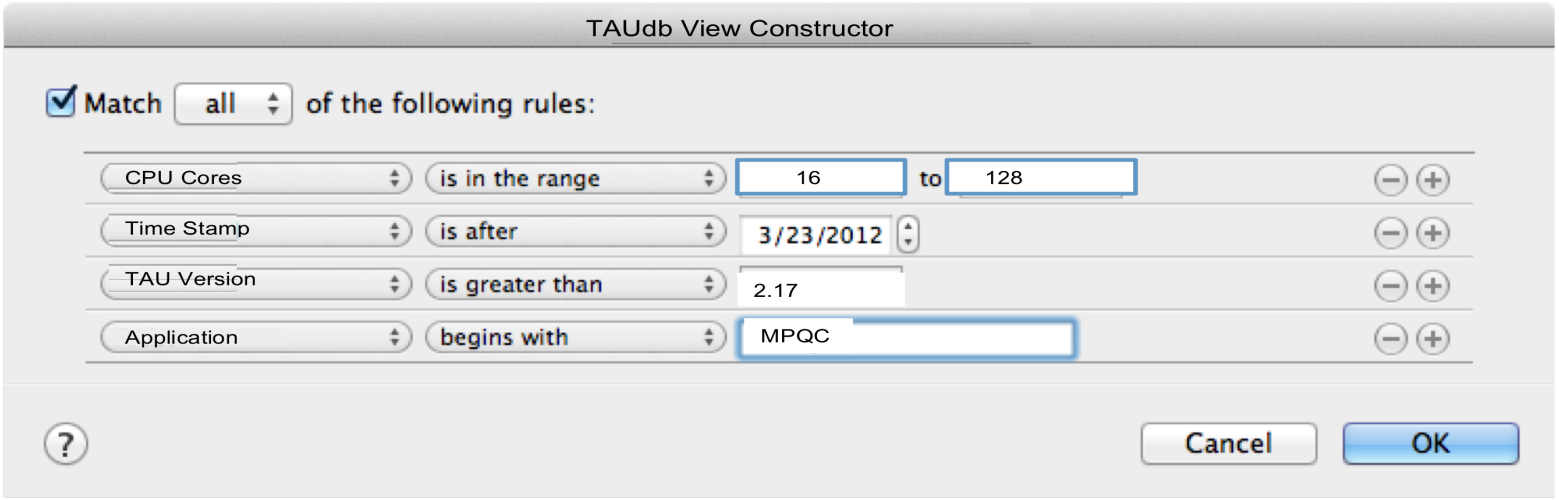
- **Application, experiment tables are gone**
- **View** (user-defined grouping/filter of Trials or Views of Trials – arbitrary depth) defined using Metadata
  - Replacement for application & experiment
- **Trial** (still a single profile)
  - **Primary_Metadata** – name : value pair, common to all threads, no hierarchical data – handles most common cases
  - **Secondary_Metadata** – could be unique for each thread, phase, timer, can be hierarchical or arrays
  - Both can be queried directly
    - View : "Give me trials of application ABC with 4096 processes which ran on machine X with dataset Y in the last 30 days"

Schema Organizational Changes



View Creation Process

- Using trial metadata name/values, filter (out) the trials of interest
- Same interface in PerfExplorer, ParaProf, web



View creation GUI mock-up

Metadata Collection

- Like PerfDMF, can load Metadata file with profile data
  - XML
  - JSON http://www.json.org
    - Like XML, allows for arbitrarily structured data
    - Less annotation overhead, no pre-defined schema
- Example:

```
{ "metadata_number" : 14,
  "metadata_string" : "string",
  "metadata_boolean" : false,
  "metadata_array" : [1,2,3],
  "metadata_null" : null,
  "metadata_object" : { "inner_object" : "value" },
  "metadata_array_of_objects" : [ { "name" : "value" }, { "next" : "value" } ] }
```

JSON Format

- JavaScript Object Notation

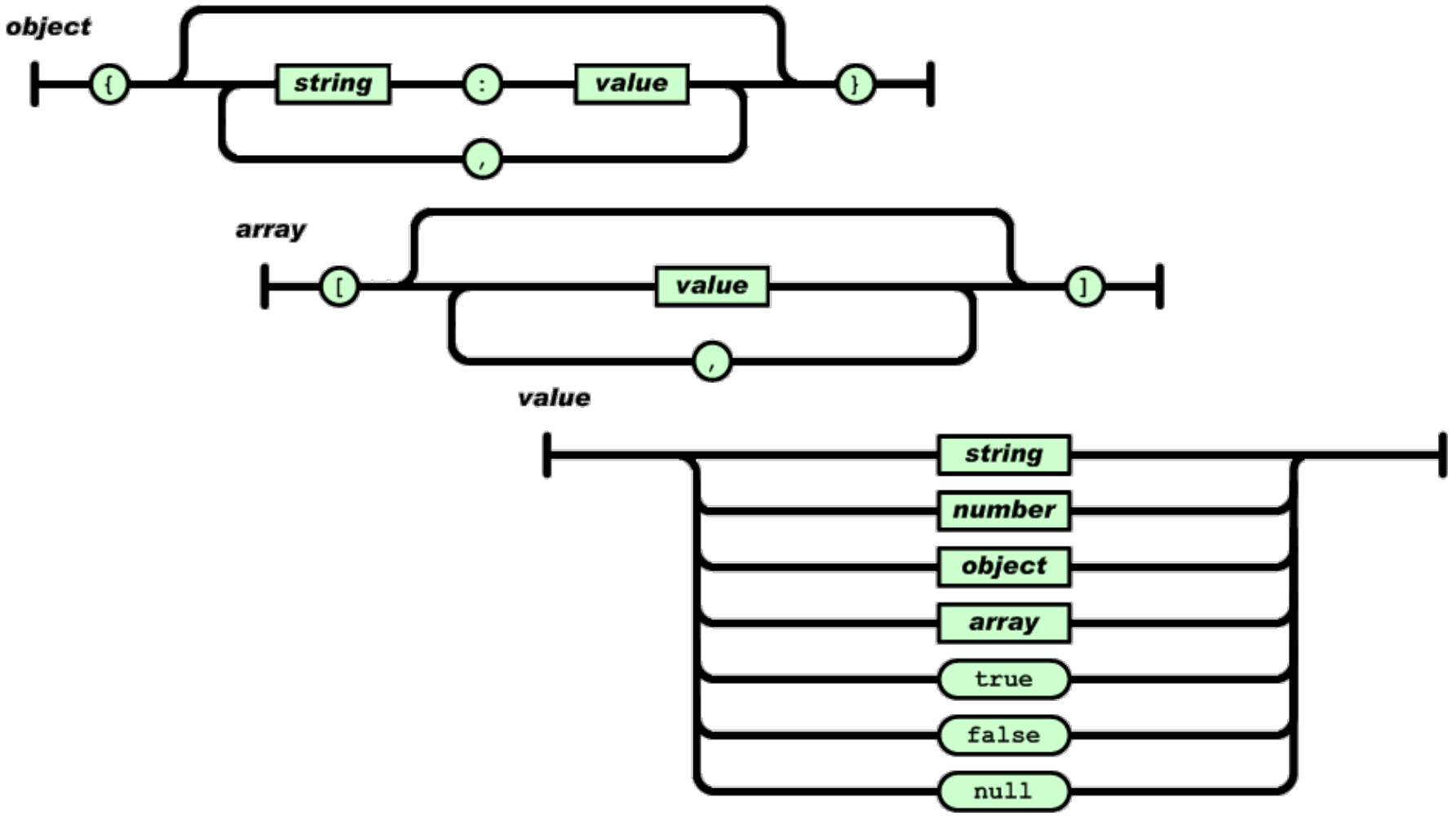
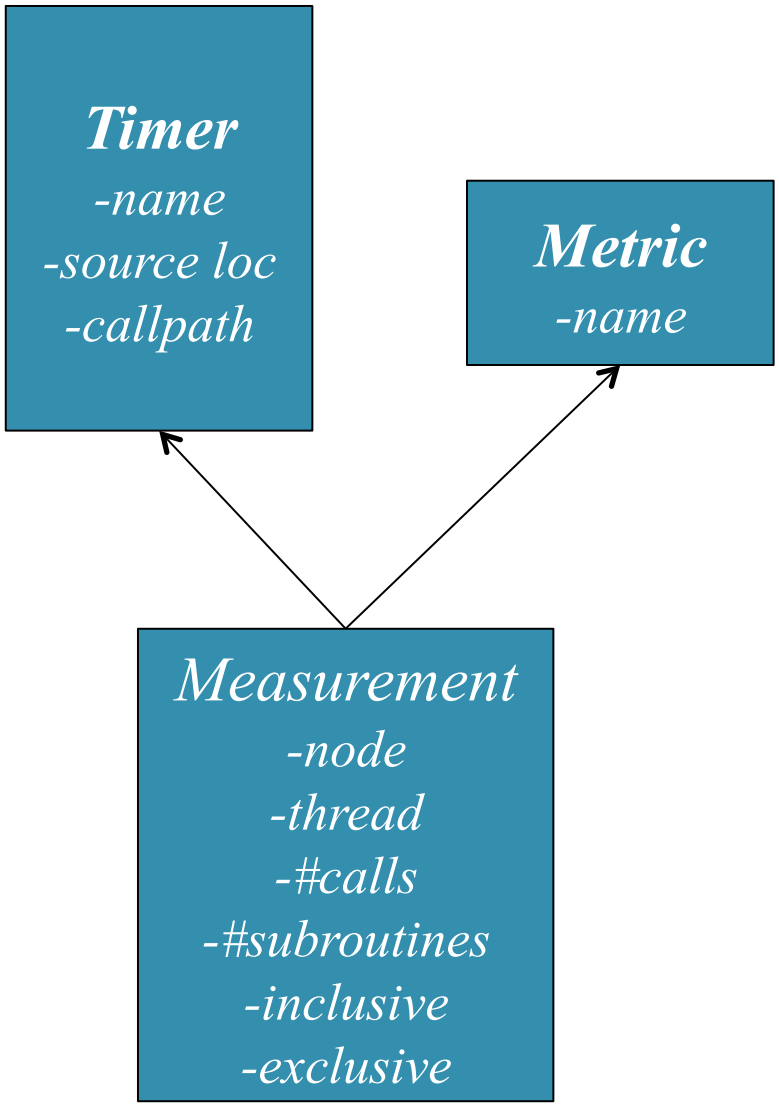


Figure credits: http://www.json.org

Profile Data Schema Changes

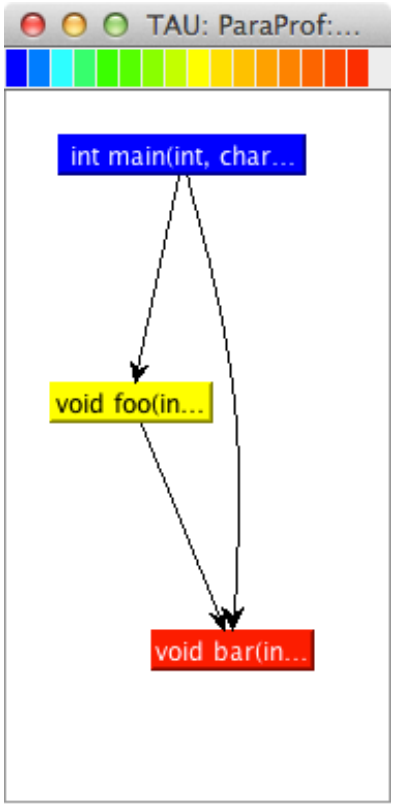


- Redundancies (source location, calls, subroutines, node/context/ thread)
- No explicit tables representing “location”, “context” or “state”
- Encoded strings (main => foo => iteration12 => bar)
- Only 3 of 5 dimensions explicitly supported

Simple Example

```
void bar (int x) {
  sleep(1);
}
void foo (int x) {
  sleep(1);
  bar(x);
}
int main (int argc, char** argv) {
  int x = 0;
  for (x = 0 ; x < 10 ; x++) {
    foo (x);
    bar (x);
  }
}
```

- What happens when we store a callpath profile of this program?



Simple example in old schema (reduced detail)

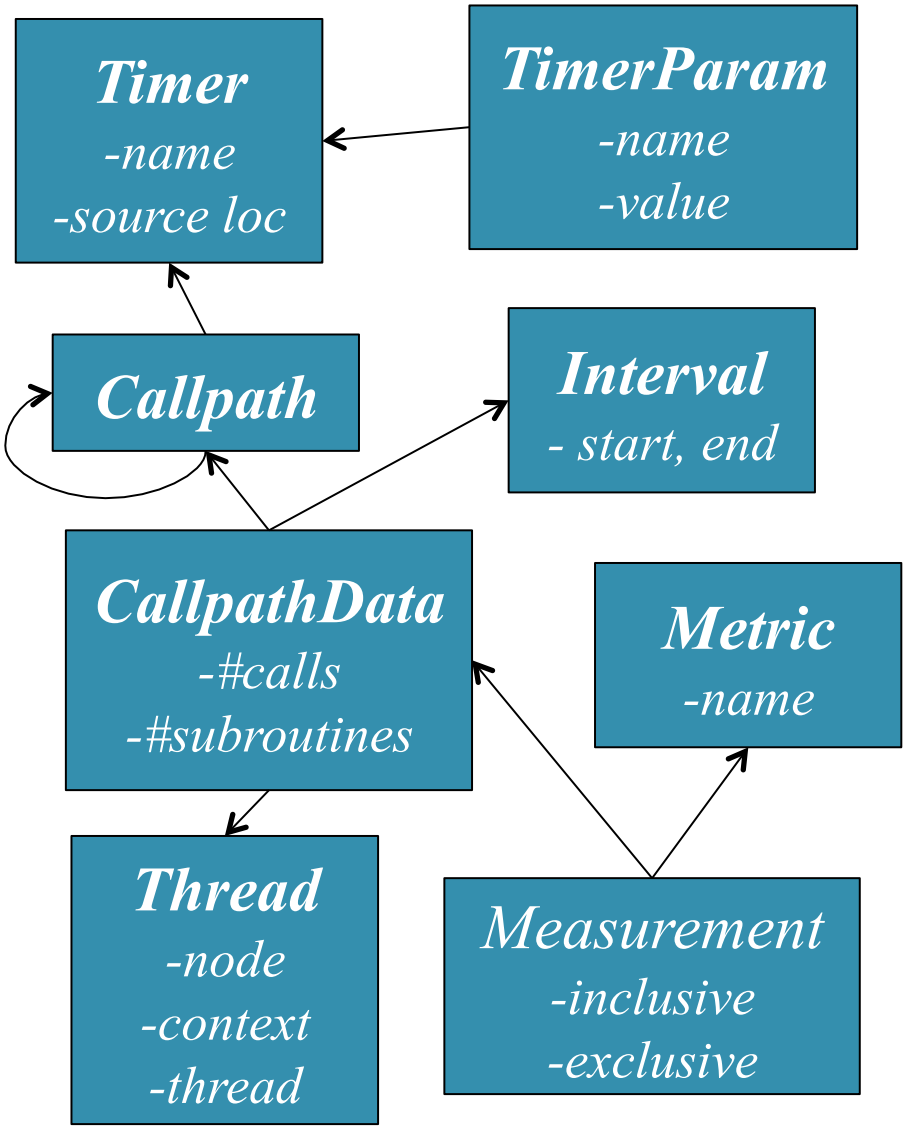
rows = callgraph nodes + callgraph edges

ID	Name	File	Line	Line end	Column	Column end
1	main()	test.c	12	18	1	1
2	foo()	test.c	7	10	1	1
3	main() => foo()	test.c	7	10	1	1
4	bar()	test.c	3	5	1	1
5	main() -> foo() => bar()	test.c	3	5	1	1
6	main() => bar()	test.c	3	5	1	1

rows = (callgraph nodes + callgraph edges) * # threads * # metrics

Timer	Node	Thread	Metric	Inclusive	Exclusive	Incl %	Excl %	Calls	Subr
1	0	0	0	30003073	49	100.00%	0.00%	1	20
2	0	0	0	20001985	10001026	66.67%	33.33%	10	10
3	0	0	0	20001985	10001026	66.67%	33.33%	10	10
4	0	0	0	20001998	20001998	66.67%	66.67%	20	0
5	0	0	0	10000959	10000959	33.33%	33.33%	10	0
6	0	0	0	10001039	10001039	33.33%	33.33%	10	0

Profile Data Schema Changes



- Redundancies eliminated
- Thread table represents location
- TimerParam table represents state
- Callpath object represents call tree context
- Normalizing the schema results in space savings (timer, metric) - ~30%
- Long names eliminated

Simple Example in New Schema

TIMER: # rows = callgraph nodes

ID	Name	File	Line	Line end	Column	Column end
1	main()	test.c	12	18	1	1
2	foo()	test.c	7	10	1	1
3	bar()	test.c	3	5	1	1

CALLPATH: # rows = callgraph nodes + callgraph edges

ID	Timer	Parent
1	1	-
2	2	-
3	2	1
4	3	-
5	3	3
6	3	1

THREAD: # rows = # threads

ID	Node	Thread
0	0	0

CALLPATH_DATA: # rows = (callgraph nodes + callgraph edges) * # threads

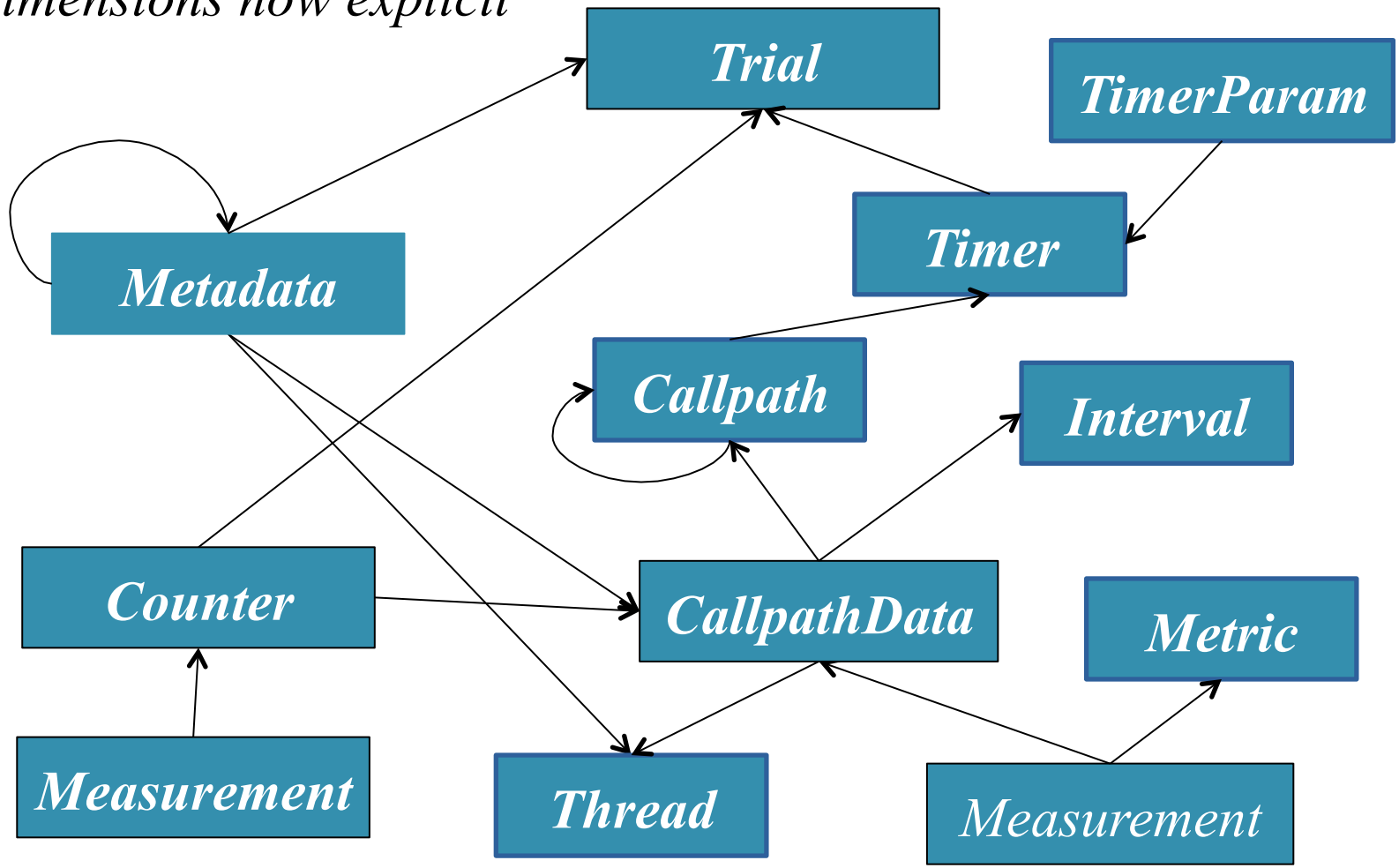
Callpath	Thread	Calls	Subr
1	0	1	20
2	0	10	10
3	0	10	10
4	0	20	0
5	0	10	0
6	0	10	0

MEASUREMENT: # rows = (callgraph nodes + callgraph edges) * # threads * # metrics

TCD	Metric	Incl	Excl	Incl %	Excl %
1	1	30003073	49	100.00%	0.00%
2	1	20001985	10001026	66.67%	33.33%
3	1	20001985	10001026	66.67%	33.33%
4	1	20001998	20001998	66.67%	66.67%
5	1	10000959	10000959	33.33%	33.33%
6	1	10001039	10001039	33.33%	33.33%

TAUdb redesigned schema

- ✓*Metadata can be queried, context-sensitive*
- ✓*All 5 dimensions now explicit*



C API

- C programming interface under development
- PostgreSQL support first, others as requested
- Prototype developed
  - Query only, both old and new schema
- Plan full-featured API: Query, Insert, & Update
- One internal test user so far – Nick Chaimov using it for Active Harmony / CHiLL work
- Request for SQLite support – currently evaluating JDBC clients

Project Status, Conclusions, Future

- New database schema defined
- C API in development
- Java API (mostly) supports the new schema
  - Supports JSON metadata (and previous XML support)
  - Not in most TAU recent release
- View construction GUI still in design phase
- TAU measurement API needs design, implementation, testing (context-sensitive metadata)
- Planning targeted distribution
  - TAUdb, ParaProf, PerfExplorer
  - Goal: zero-step config (likely 1-step config)

Support Acknowledgements

- Department of Energy (DOE)
  - Office of Science
  - ASC/NNSA
  - SUPER project
- Department of Defense (DoD)
  - HPC Modernization Office (HPCMO)
- NSF Software Development for Cyberinfrastructure (SDCI)
- Research Centre Juelich
- Argonne National Laboratory
- Technical University Dresden
- ParaTools, Inc.
- NVIDIA

ParaTools