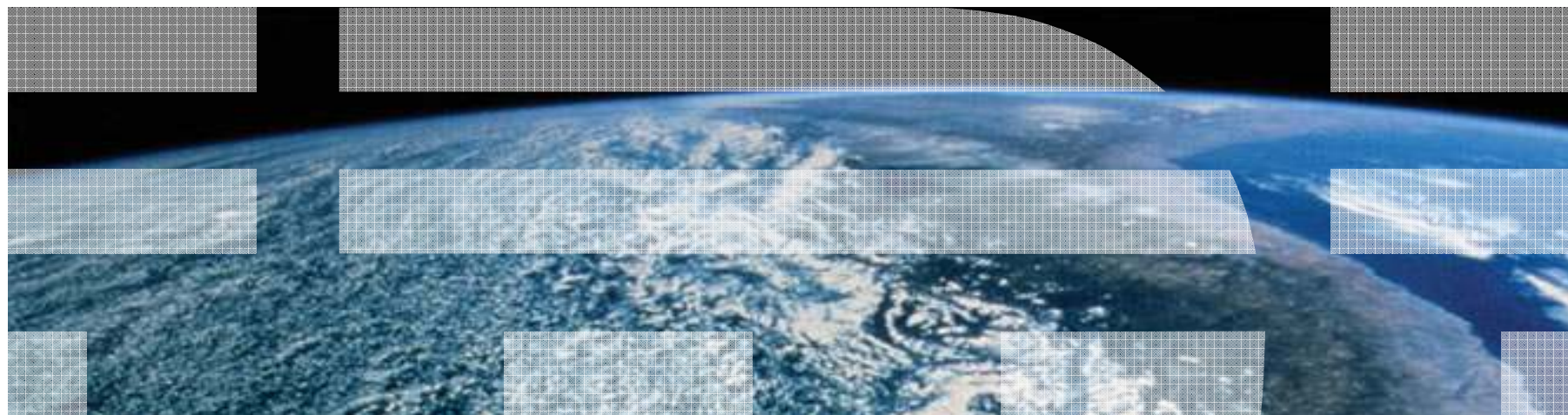

Tracing Cloud Workloads for Processor Design Evaluations

Aug 20, 2015

Saritha Vinod (IBM)
Biplob Mishra (IBM)



- Emerging Workloads and Next Gen Processor Design
- Traces and Hardware Model
- Key Challenges in Workload Tracing
- Case-study of Cloud Data Store
- Addressing the Challenges in Trace Collection and Analysis
 - When to Take a Trace ?
 - How to Select a Trace Segment ?
 - How is Trace Generated?
 - Trace Validation and Profile Analysis
- Summary

- New business trends leading to emerging workloads in domains such as big data, analytics, cloud, social and mobile

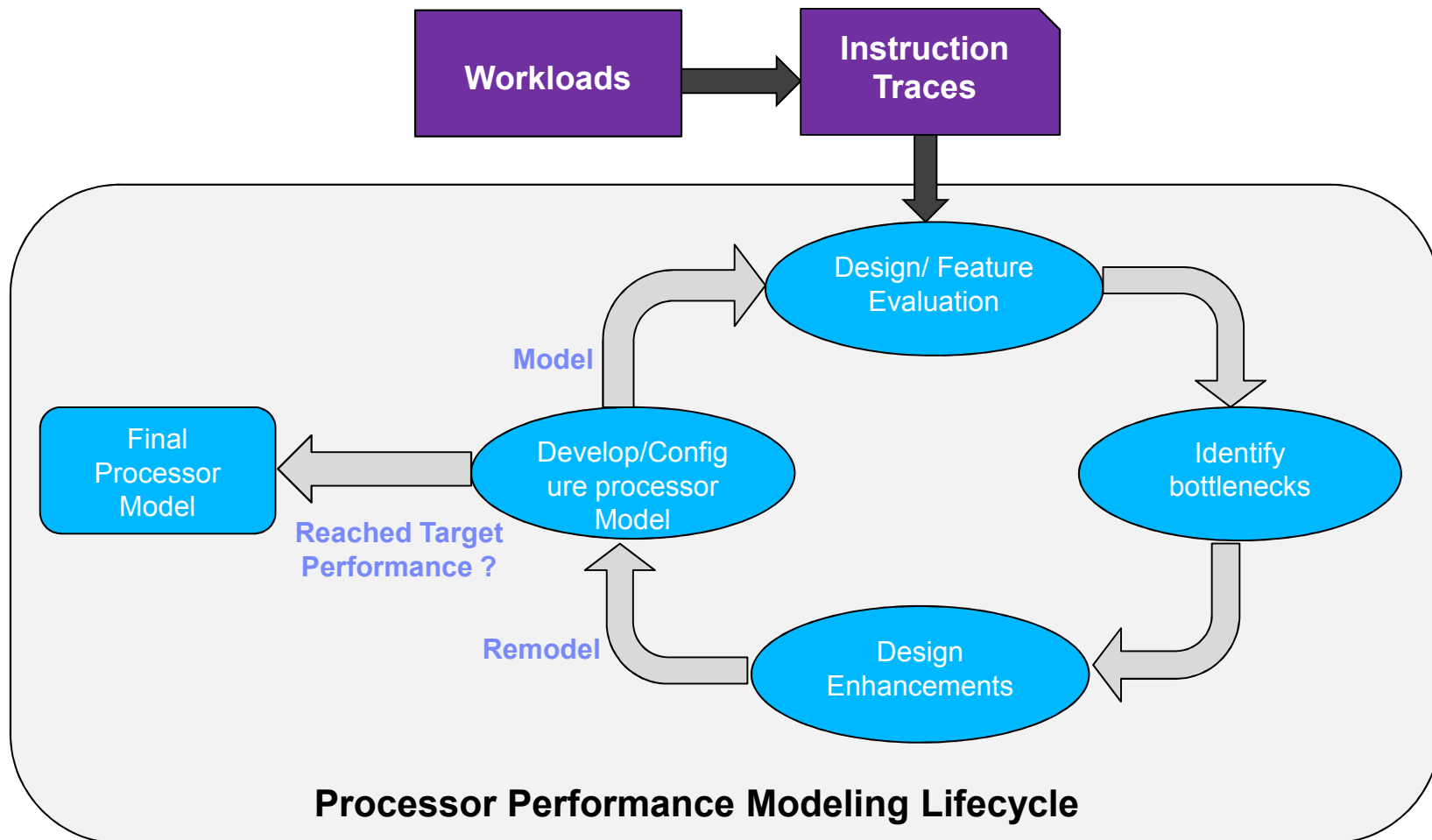


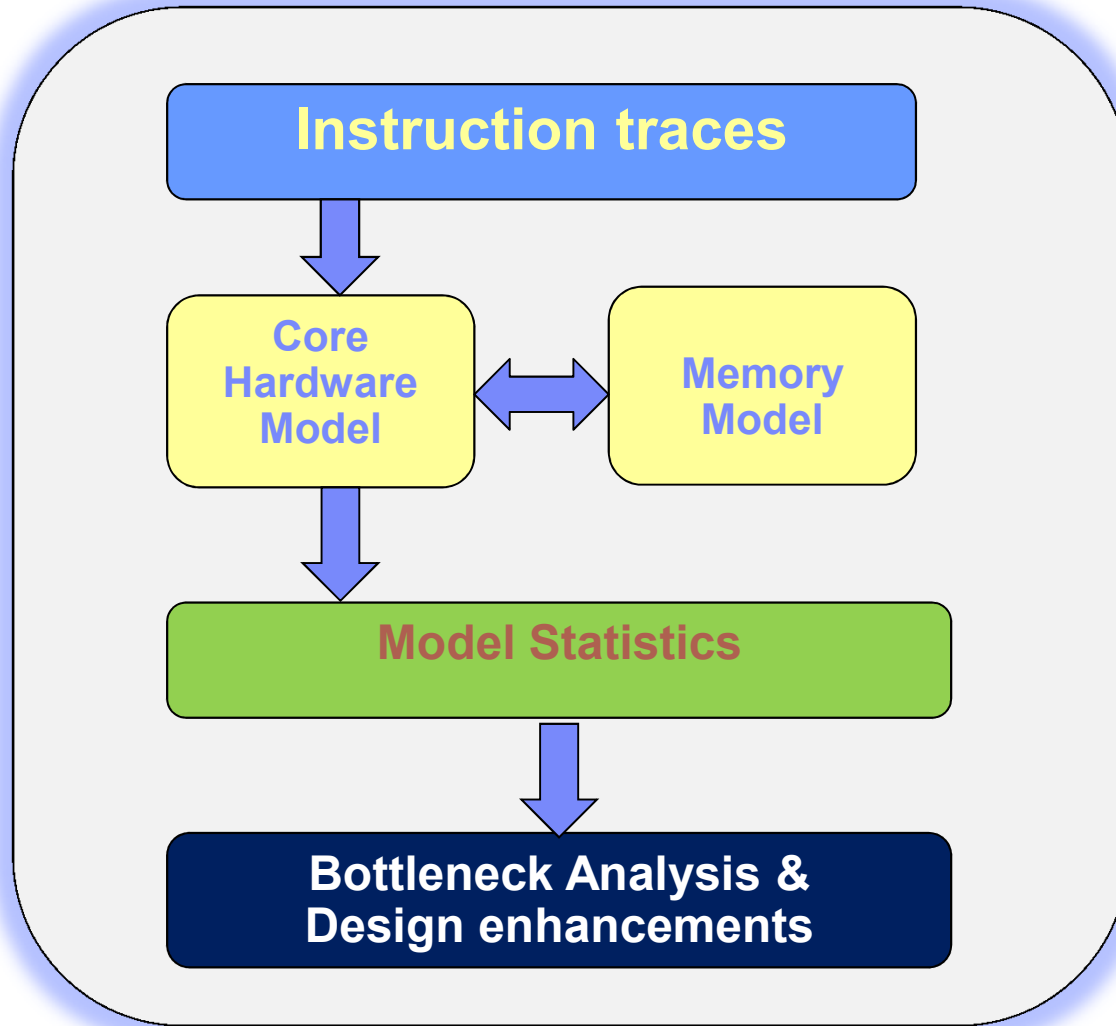
- It is important to focus on the following emerging workload characteristics while designing next generation processors
 - Instruction combinations
 - Cache access patterns
 - Data access patterns
 - Sharing of data
 - OS and Hypervisor calls
 - Instruction mixes
 - Data affinities
 - Branch related instruction mix
- How are these workload characteristics taken in as input to the processor design lifecycle ? ***Through workload traces***

Processor Performance Modeling Lifecycle

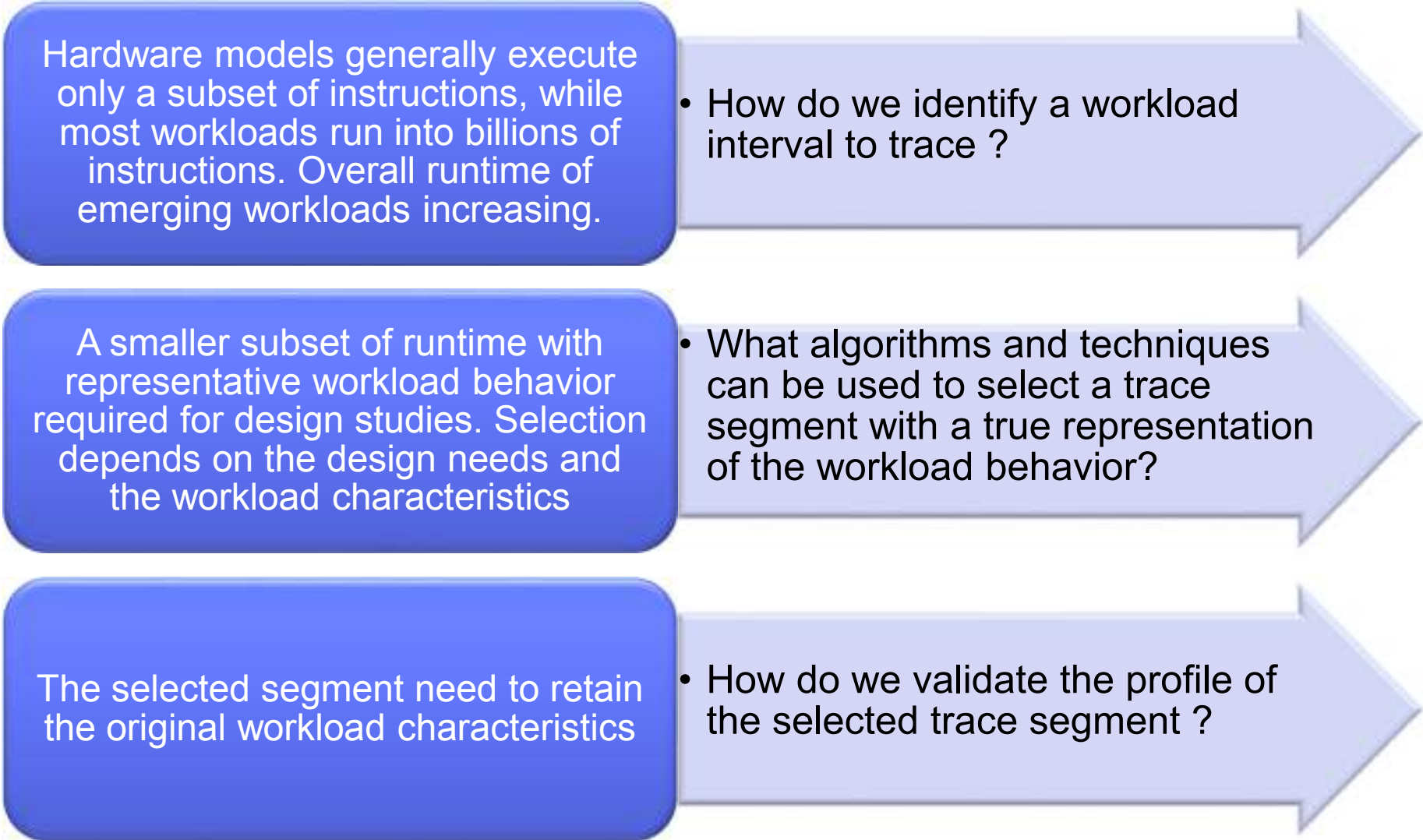


- ❑ To achieve best performance for emerging workloads the next generation processor design need to use appropriate workload traces

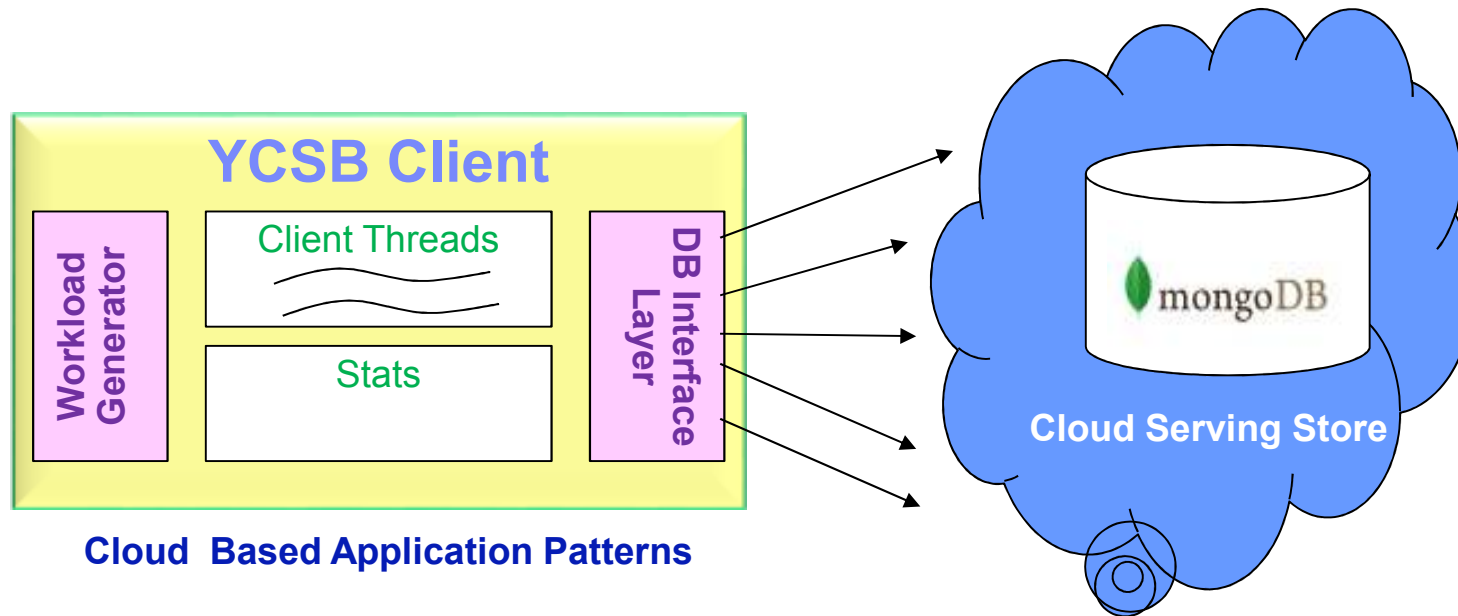




Key Challenges in Workload Tracing



Case-study of Cloud Data Store (MongoDB) Driven by a Cloud Benchmark (YCSB)



Cloud Based Application Patterns

- Single MongoDB instance
 - Focus on transaction performance than scalability
- Driven through Yahoo Cloud Serving Benchmark (YCSB) Client
 - Workload generator modeling applications on cloud data system

- No-SQL, Document-oriented
- High performance, High availability, Easy scalability
- Clients connect to the MongoDB server process and perform a sequence of actions, such as inserts, queries and updates
- Stores data in files and uses memory mapped files for efficient data management
- MongoDB has multiple use cases. Analyze use cases to generate similar workload pattern

MongoDB Use Cases

High Volume Data Feeds

Operational Intelligence

User Data Management

Content Management

Product Data

Configures Cloud Workload pattern

- With a particular mix of read/write operations, data sizes and request distributions
- Used for modeling fundamental workloads generated by web applications on cloud data system

Provides Performance Benchmark Tier

- Focuses on the latency of requests when the database is under load, using constant hardware resources (scaling not considered in this tier)

Acts as Data and Workload Generator

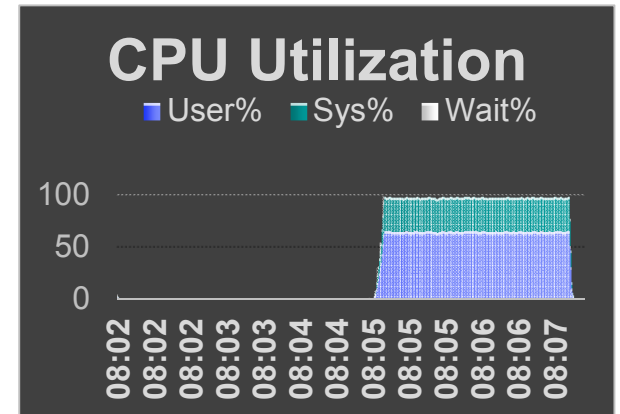
- Defines the dataset and load it into the database (Load Phase)
- Executes operations against the dataset while measuring performance (Transaction Phase)

Addressing the Challenges in Trace Collection and Analysis

When to Take a Trace ?



- Wait for application steady state

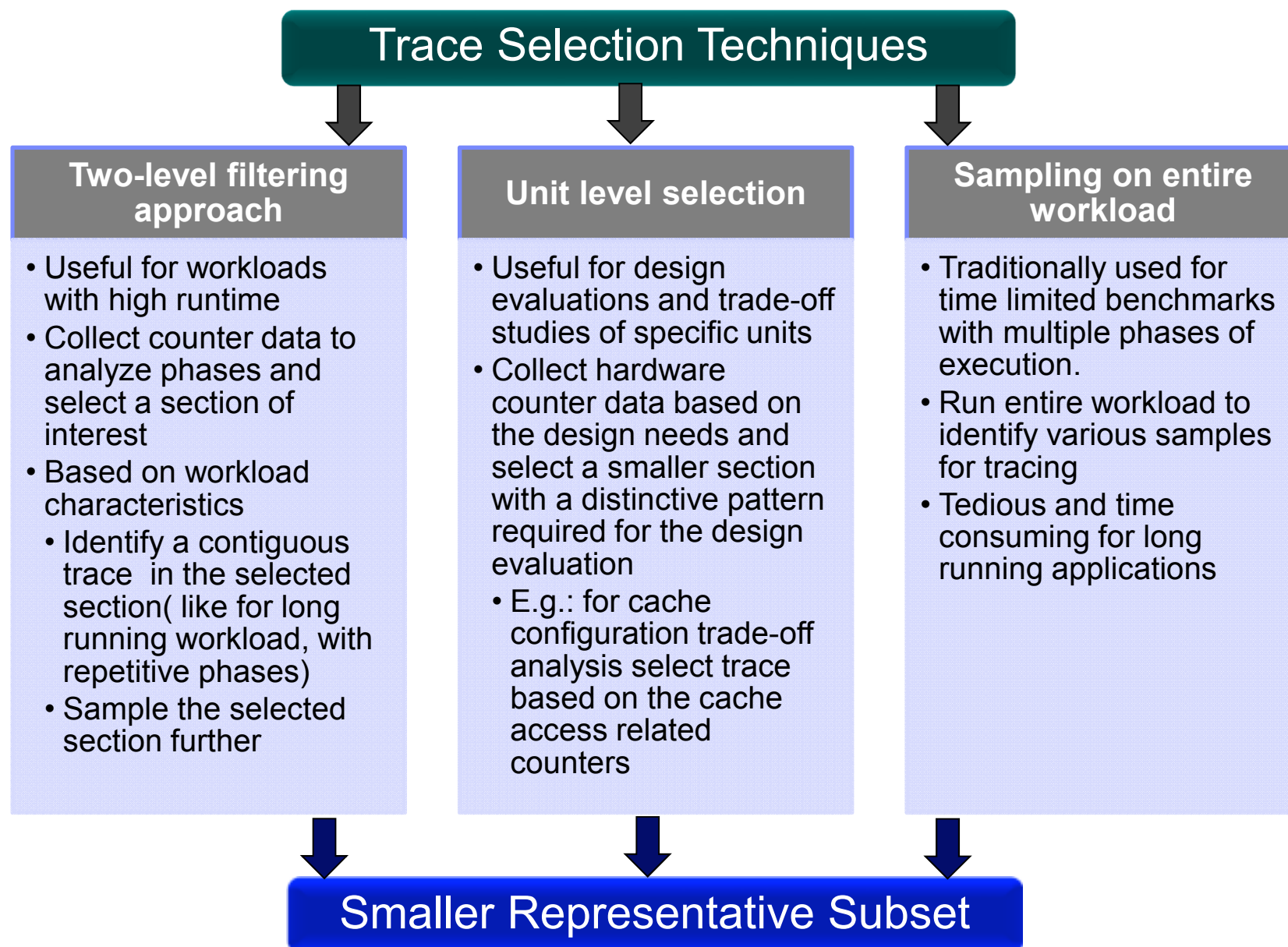


CPU utilization

Application transaction pattern. E.g.: number of records processed, throughput, latency, etc.

Application profile using hardware performance counters. Eg: memory bandwidth, cache misses etc.

How to Select a Trace Segment ?

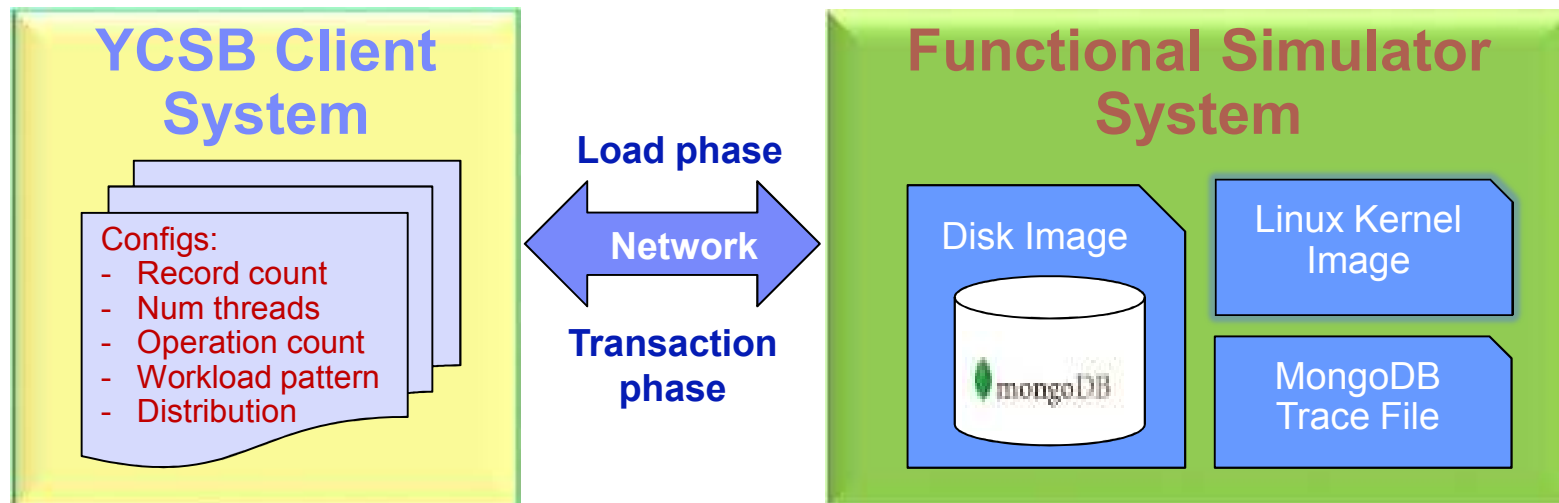


How is Trace Generated?



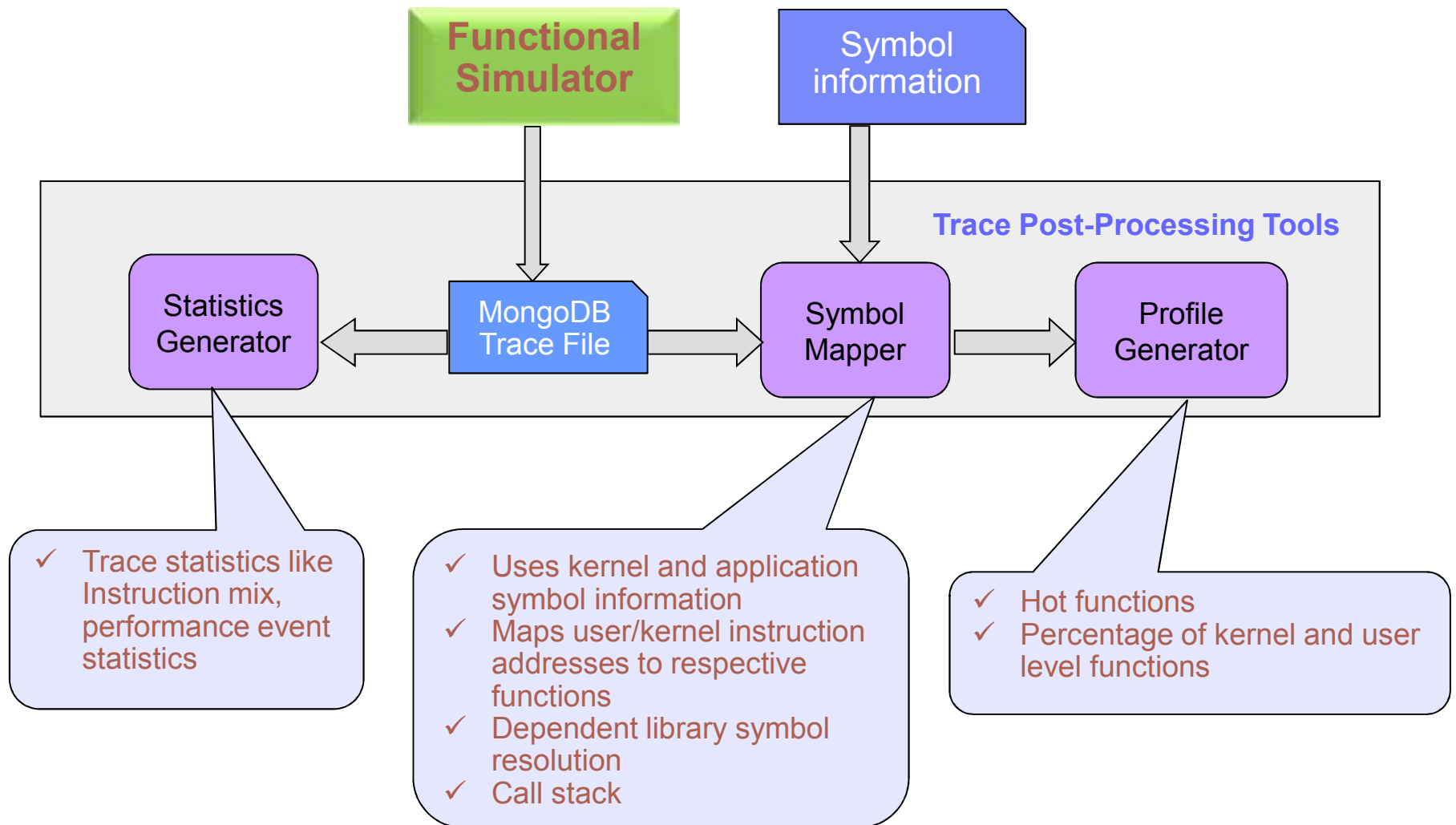
Setup the functional simulator to trace the selected representative segment

MongoDB Workload Tracing Setup



- Load phase happens first, followed by Transaction phase
- Trace only the transaction phase
- Load phase mostly repetitive in nature; Transaction phase used for performance measurement, will have interesting sequence of instructions

Trace Validation and Profile Analysis



Symbol Mapper – A sample output



- Understand the workload runtime that was captured in the trace.

Kernel instructions

```
152598 0xc000000007ade58 - 0x60000000 - .....tcp_recvmsg <+840> <-2122> <+840> <27> <0x19a274>
152599 0xc000000007e082c - 0xe8410018 - .....inet_recvmsg <+124> <-102> <+124> <9> <0x19a27e>
152600 0xc0000000073b970 - 0xe8410018 - .....sock_recvmsg <+160> <-114> <+160> <11> <0x19a28a>
152601 0xc0000000073e7c0 - 0x2f830000 - .....sys_recvfrom <+224> <-242> <+224> <8> <0x19a293>
152602 0xc000000001f3808 - 0x7c0802a6 - .....fput <+8> <-250> <+8> <18> <0x19a2a6>
152603 0xc0000000073e844 - 0x60000000 - .....sys_recvfrom <+356> <-110> <+356> <13> <0x19a2b4>
152604 0xc0000000073f630 - 0x7c6307b4 - .....sys_socketcall <+672> <-322> <+672> <7> <0x19a2bc>
152605 0xc00000000090d8 - 0xf86101c8 - .....syscall_exit <+0> <-154> <+0> <36> <0x19a2e1>
152606 0x106bb87c - 0xe8410018 - ..mongo::Socket::_recv <+76> <-38> <+76> <5> <0x19a2e7>
152607 0x106bb8c0 - 0x60000000 - ...mongo::Socket::unsafe_recv <+32> <-50> <+32> <8> <0x19a2f0>
152608 0x106bdf1c - 0x60000000 - ...mongo::Socket::recv <+140> <-438> <+140> <26> <0x19a30b>
152609 0x106b3d98 - 0x60000000 - .....mongo::MessagingPort::recv <+296> <-1666> <+296> <24> <0x19a324>
152610 0x106b77f4 - 0x60000000 - .....mongo::PortMessageServer::handleIncomingMsg <+1044> <-2274> <+1044> <14> <0x19a333>
152611 0x101b1240 - 0x3c401098 - .....mongo::MyMessageHandler::process <+0> <-2302> <+0> <41> <0x19a35d>
152612 0x103ecb28 - 0x60000000 - .....mongo::inShutdown <+8> <-38> <+8> <5> <0x19a363>
152613 0x101b12e8 - 0x60000000 - .....mongo::MyMessageHandler::process <+168> <-2134> <+168> <10> <0x19a36e>
152614 0x104228c8 - 0x7c0802a6 - .....mongo::LastErrorHolder::startRequest <+8> <-106> <+8> <8> <0x19a377>
152615 0x104227c8 - 0x7c0802a6 - .....mongo::prepareErrForNewRequest <+8> <-238> <+8> <22> <0x19a38e>
152616 0x104228ec - 0x60000000 - .....mongo::LastErrorHolder::startRequest <+44> <-70> <+44> <6> <0x19a395>
152617 0x101b1314 - 0x60000000 - .....mongo::MyMessageHandler::process <+212> <-2090> <+212> <13> <0x19a3a3>
152618 0x106b3a30 - 0x3c401098 - .....mongo::MessagingPort::remote <+0> <-338> <+0> <20> <0x19a3b8>
152619 0x106b3afc - 0xe8410018 - .....mongo::MessagingPort::remote <+204> <-134> <+204> <16> <0x19a3c9>
152620 0x101b134c - 0xe8410018 - .....mongo::MyMessageHandler::process <+268> <-2034> <+268> <4> <0x19a3ce>
152621 0x103f5228 - 0x7c0802a6 - .....mongo::assembleResponse <+8> <-7814> <+8> <40> <0x19a3f7>
152622 0x103f5770 - 0xe8410018 - .....mongo::assembleResponse <+1360> <-6462> <+1360> <13> <0x19a405>
152623 0x1058d398 - 0x7c0802a6 - .....mongo::OpCounters::gotOp <+8> <-570> <+8> <44> <0x19a432>
152624 0x103f52f0 - 0x60000000 - .....mongo::assembleResponse <+208> <-7614> <+208> <9> <0x19a43c>
```

Application instructions

- A faster drift in the newer workloads, more and more new traces need to be collected. Quality of traces and faster analysis important.
- Trace selection techniques based on a wide variety of hardware needs and workload characteristics
- Trace Validation process is gaining lot of significance; emerging workloads with complex runtime and multiple components in the application stack
- Multiple choices for trace selection:
 - Sampling on entire workload to identify phases
 - Two level filtering and contiguous trace selection (for longer runtime, transaction based workloads etc.)
 - Unit level selection (for design evaluations of specific processor features)

Questions

