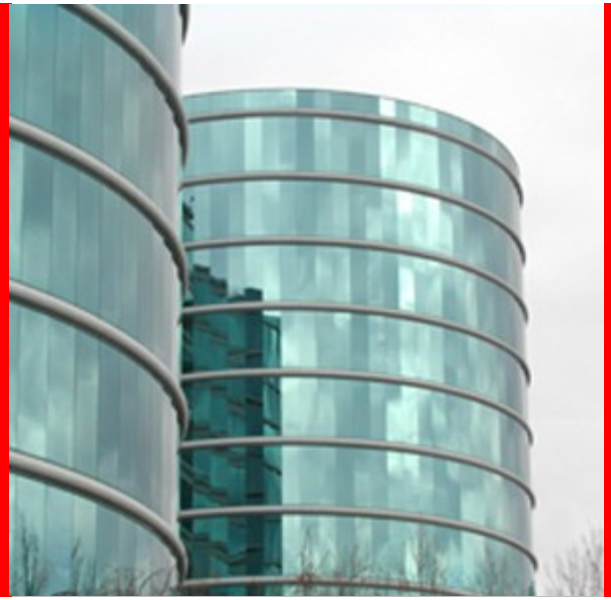

ORACLE®



ORACLE

The Other HPC: Profiling Enterprise-scale Applications

Marty Itzkowitz
Senior Principal SW Engineer, Oracle
marty.itzkowitz@oracle.com

Agenda

- HPC Applications
  - Traditional HPC
  - The Other HPC
- Profiling Enterprise-Class Applications
  - SPECjbb, SPECjAppserver, SPECjEnterprise
  - SOA
  - Oracle Database

Traditional HPC

- Intensive numerical calculations
  - Fortran/C/C++
  - OpenMP/MPI
- Run on many CPUs, nodes
  - Many threads (OpenMP)
  - Many processes (MPI)
  - Hybrid runs
- Multiple processes tend to be uniform
- Computations are mostly loop-based

The Other HPC

- Transactions and web services
  - Java/C/C++
  - Ad hoc parallelism
- Also run on many CPUs, nodes
  - Long duration — web servers run forever
  - Many threads
  - Many processes
  - But not quite peta-scale (yet)
- Multiple processes are not uniform
- Often not loop-based

Profiling Enterprise-Class Applications

- Many processes, many threads; long duration
  - Need to track all
  - Typically have long initialization phase
- Multi-thread performance issues
  - Lock contention: lock-global vs. lock-local
    - Synchronization tracing (use `collect -s on`)
    - Key issue: scoping of locks
  - Load imbalance
    - Useful work matters, not CPU usage
      - Busy-waits use CPU resources, but are not useful work

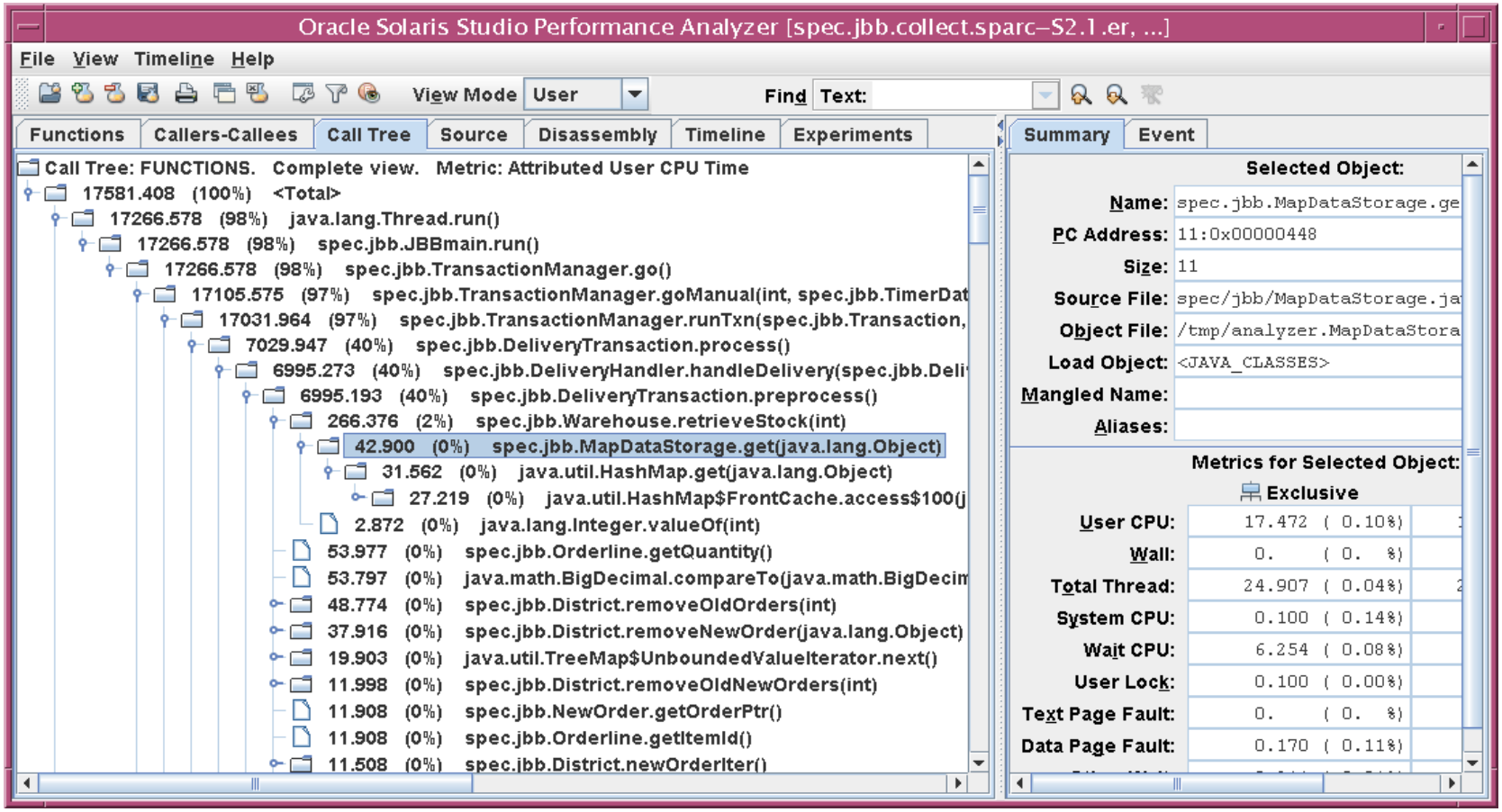
Profiling Enterprise-Class Applications (continued)

- Complex start up: launch by script
  - Add env.var. to prepend **`collect`** command to target invocation
    - No effect if not set; data collection if set
  - **`-y`** argument for data-collection control (*e.g.*, skip initialization)
  - **`-l`** argument for event marking (*e.g.*, mark transaction begin/end)
    - API calls in user code can be used to for markers, too
      - Calls ignored if no data being collected
- Filtering to drill down on problems
  - Based on function on stack
  - Based on threads, processes, CPUs
  - Between marked events

SpecJBB

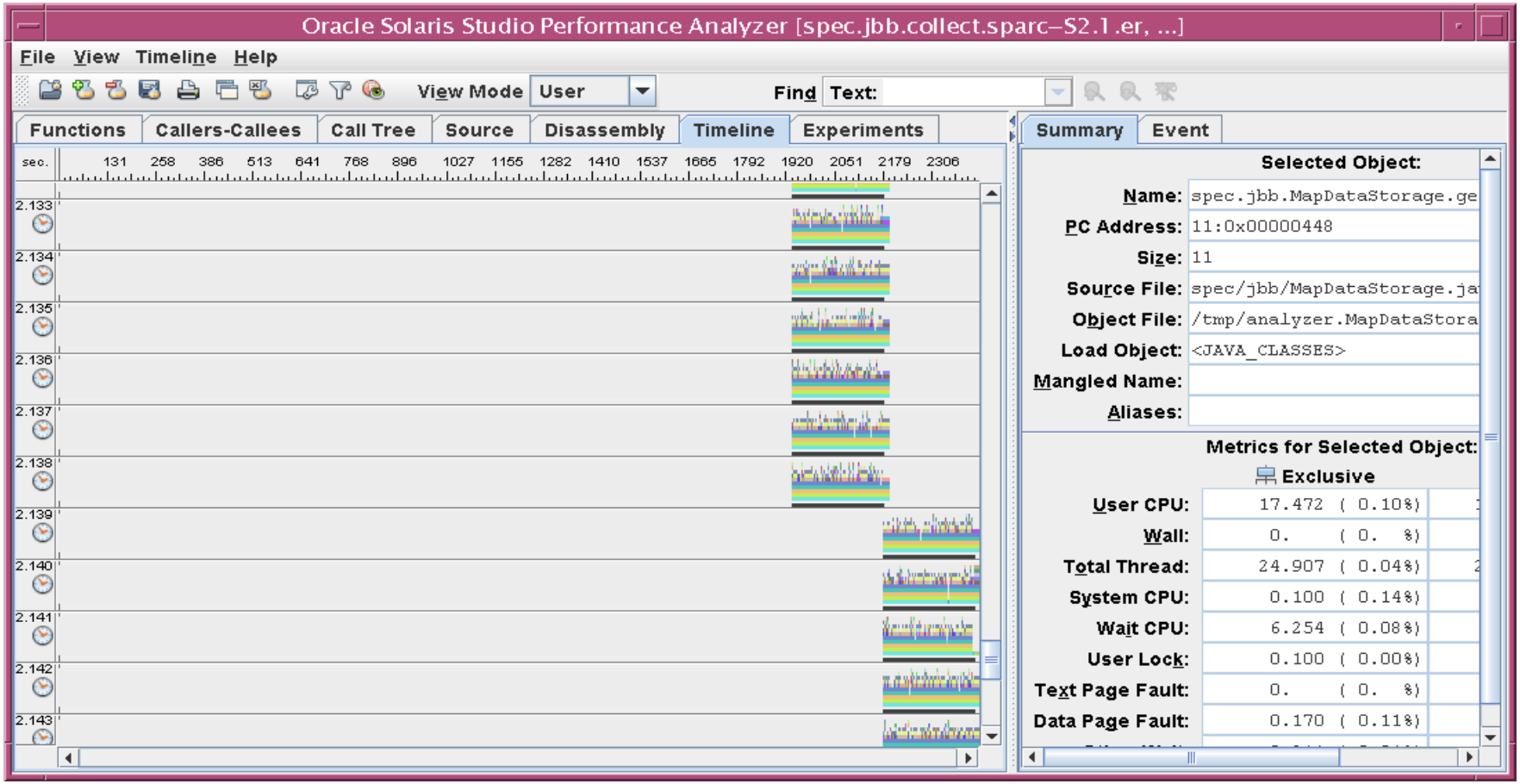
- Benchmark for three-tier enterprise system
  - Based on TPC-C
  - A small enterprise-scale application
- Models a wholesale company and order-entry system
  - Has warehouses that serve districts
    - Run does first 1, then 2, …, 16 warehouses
      - Up to twice the number of CPUs detected
      - First eight ignored, last eight count for score
  - Processes orders, deliveries, payments, *etc*.
  - Has no real database interactions
    - Data records stored as HashMaps or TreeMaps
- Run on 8-CPU machine, uses 156 threads
  - New set of 2N threads created for warehouse N
  - Completely CPU-bound

SpecJBB: Call Tree



Shows hottest path

SpecJBB: Timeline



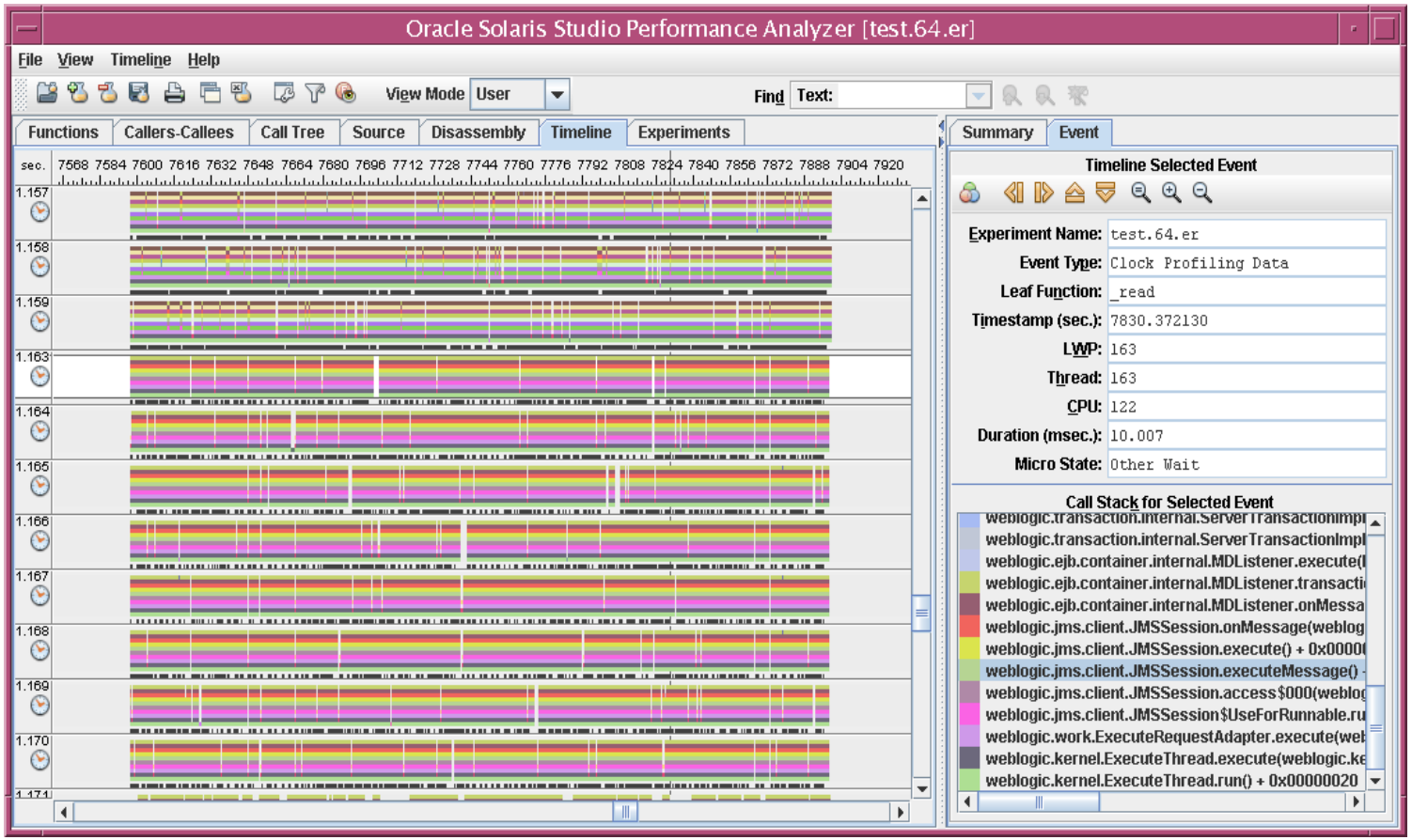
Transition from 15 warehouses to 16

Old threads terminate; new threads are created

SpecJAppServer

- Profile of WebLogic Application Server
  - Simulates standard e-commerce application
  - Processes requests from clients via browser for purchases
  - Processes requests via CORBA/IIOP to manage inventory
- Run on 128-CPU machine, uses ~280 threads
- Data collection paused during initialization phase
  - Recorded data shows active window ~400 seconds

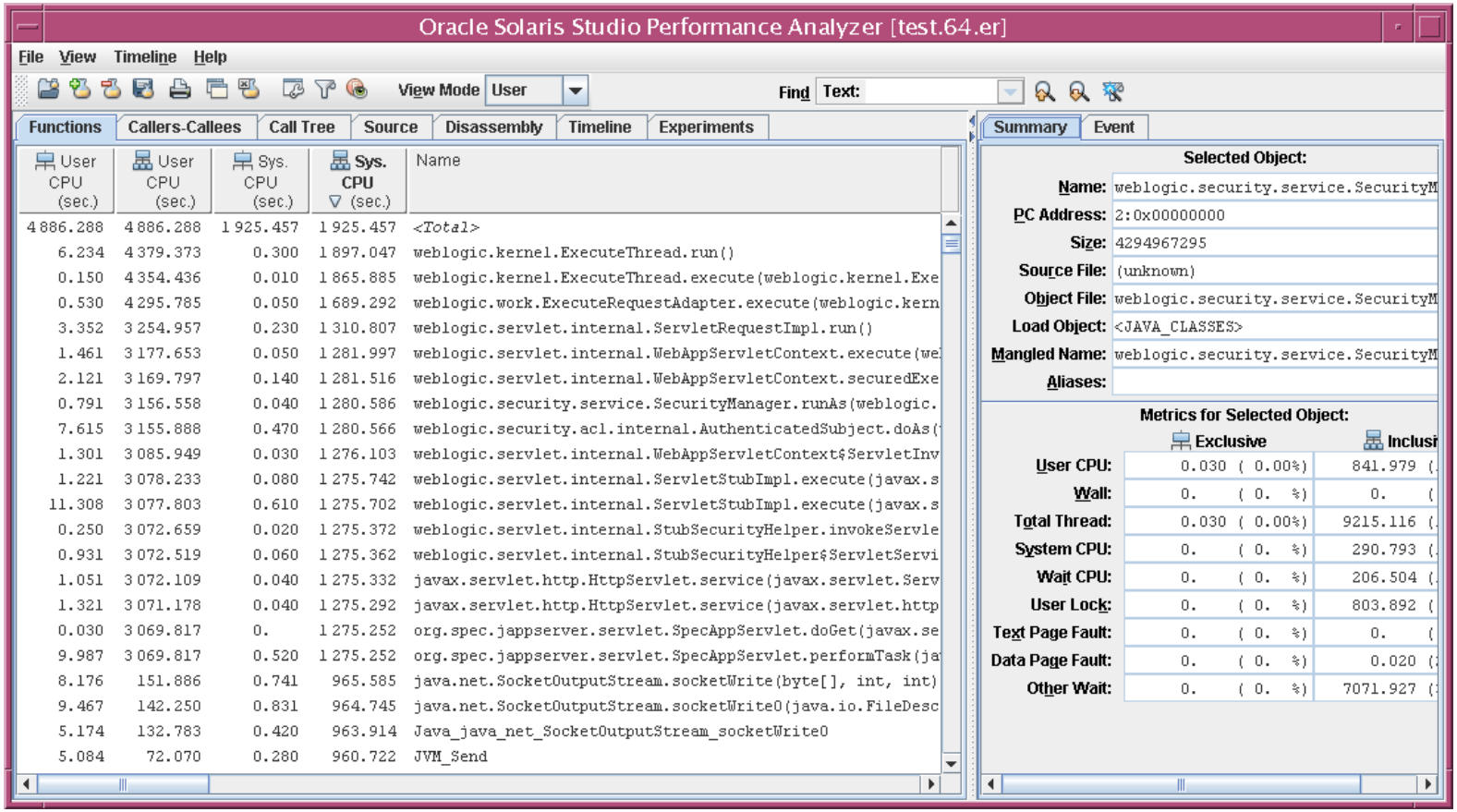
SpecJAppServer: Timeline



Time from ~7500 – 7900 seconds

Threads 157-170; two different types of threads shown

SpecJAppServer: Function List

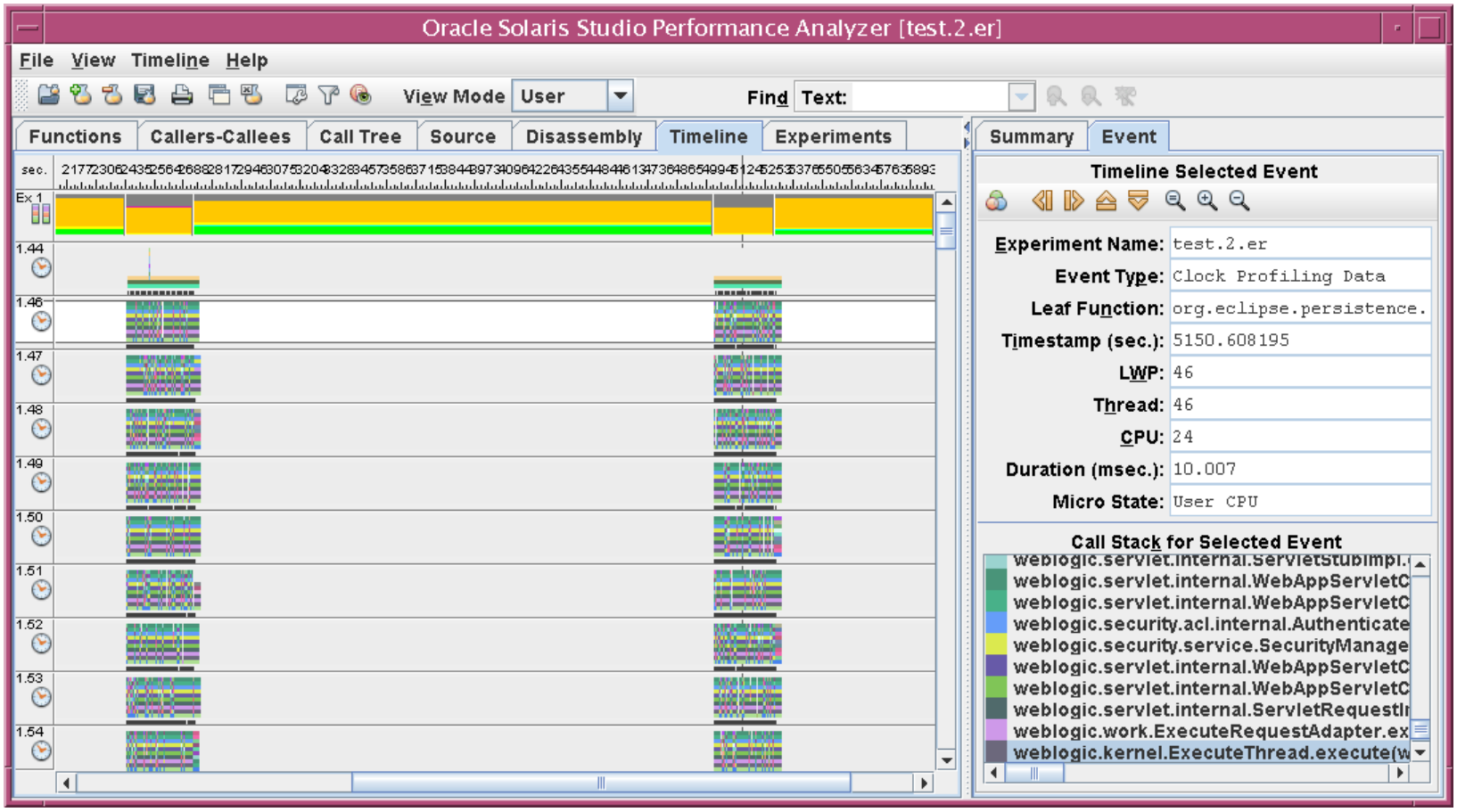


Sorted by system CPU time – implies I/O activity

SpecJEnterprise

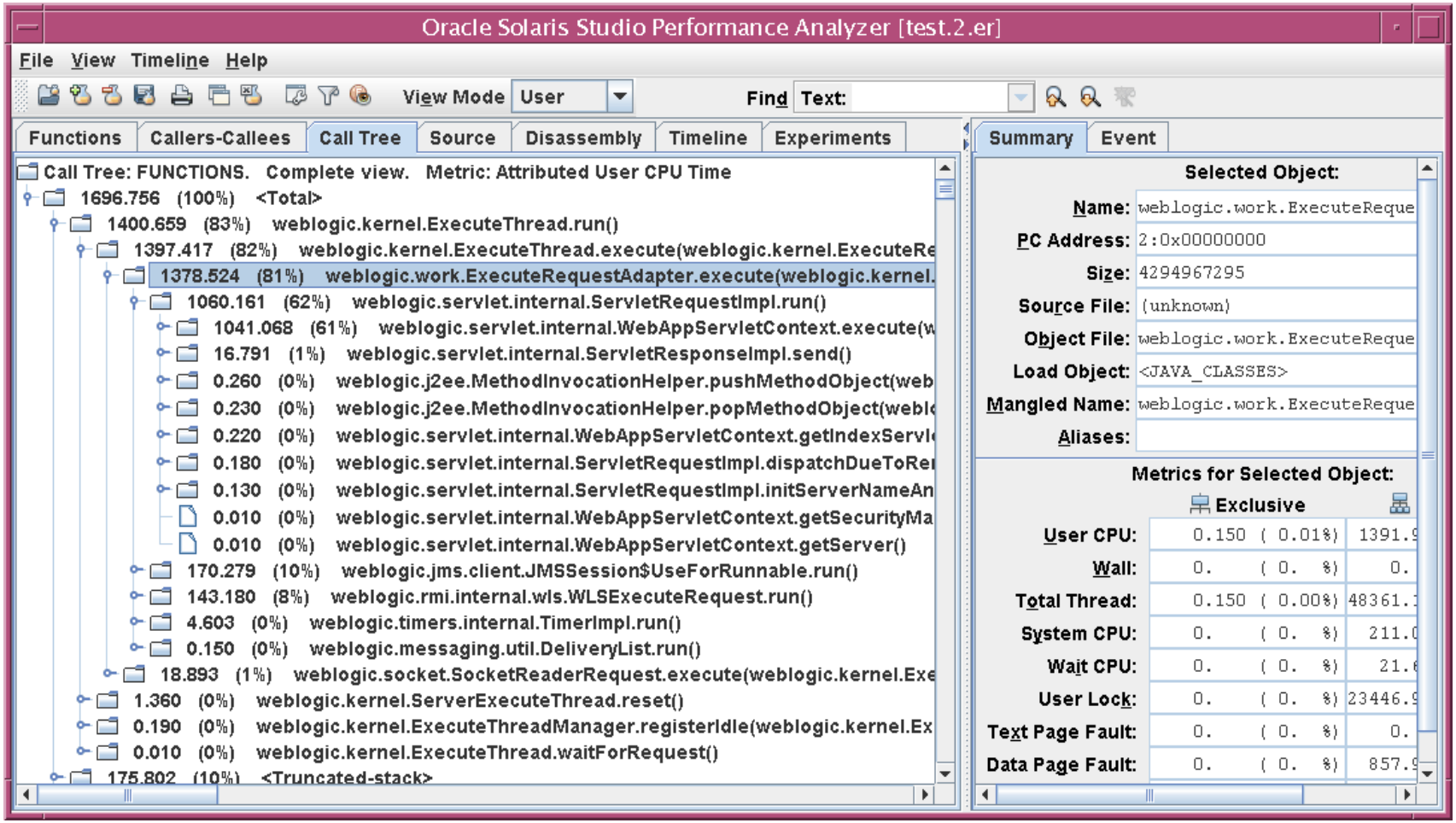
- Benchmark emulates automobile manufacturer
  - Stresses Java EE 5 servers, JVM, CPU, *etc*.
  - Three domains: Dealer, Manufacturing and Supplier
  - Driver drives the benchmark
    - Runs on different system
  - Successor benchmark to SPECjAppserver
- Run on 128-CPU machine, uses 282 threads
- Data collection enabled for two 300 second snaps
  - First at 2436 seconds, second at 5026 seconds
  - Data covers only those two intervals

SpecJEnterprise: Timeline



Data was collected only for two intervals

SpecJEnterprise: Call Tree

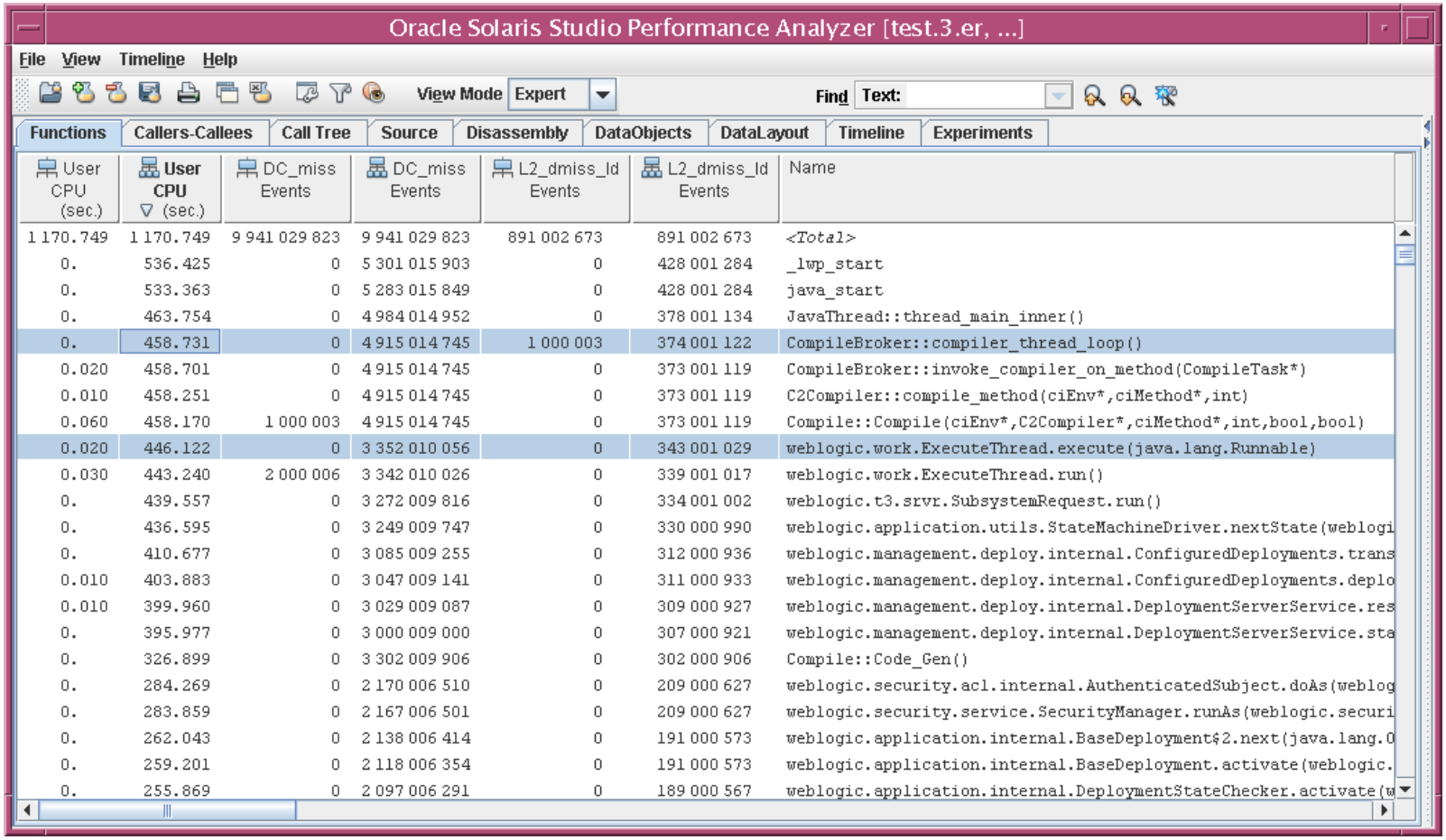


Most time spent in WebLogic middleware

Oracle SOA Suite

- SOA = Service-Oriented Architecture
- Single service component architecture
  - Based on Fusion Middleware and WebLogic
  - High throughput, low latency
  - Unified event-driven and service-oriented capabilities
    - Handles complex events
- Near real-time performance requirement
- Run on 64-CPU machine, using 166 threads
  - One run, collected clock- and cache-miss-profiles

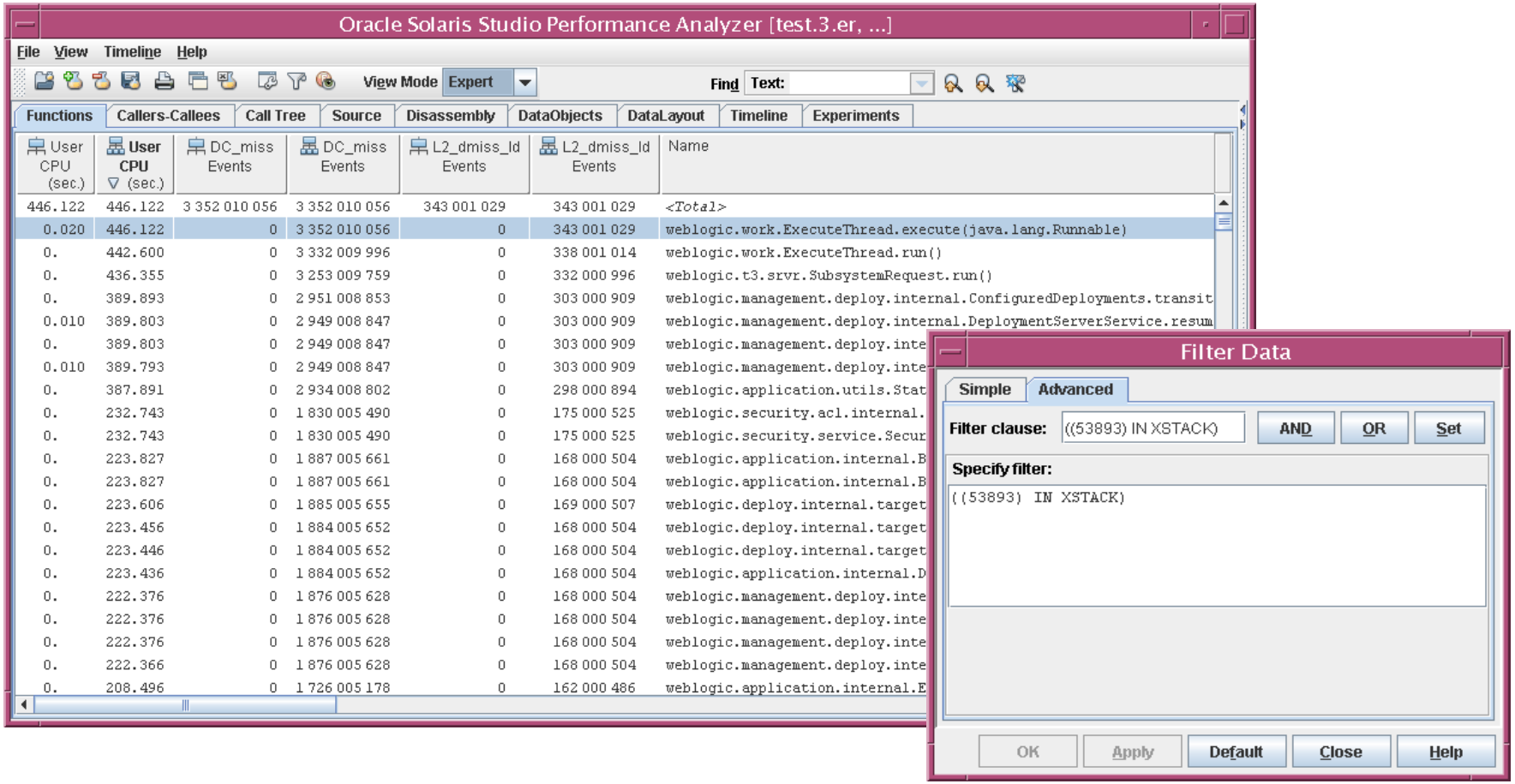
SOA: Functions



User CPU (sec.)	User CPU (sec.) ▽	DC_miss Events	DC_miss Events	L2_dmiss_ld Events	L2_dmiss_ld Events	Name
1 170.749	1 170.749	9 941 029 823	9 941 029 823	891 002 673	891 002 673	<Total>
0.	536.425	0	5 301 015 903	0	428 001 284	_lwp_start
0.	533.363	0	5 283 015 849	0	428 001 284	java_start
0.	463.754	0	4 984 014 952	0	378 001 134	JavaThread::thread_main_inner()
0.	458.731	0	4 915 014 745	1 000 003	374 001 122	CompileBroker::compiler_thread_loop()
0.020	458.701	0	4 915 014 745	0	373 001 119	CompileBroker::invoke_compiler_on_method(CompileTask*)
0.010	458.251	0	4 915 014 745	0	373 001 119	C2Compiler::compile_method(ciEnv*,ciMethod*,int)
0.060	458.170	1 000 003	4 915 014 745	0	373 001 119	Compile::Compile(ciEnv*,C2Compiler*,ciMethod*,int,bool,bool)
0.020	446.122	0	3 352 010 056	0	343 001 029	weblogic.work.ExecuteThread.execute(java.lang.Runnable)
0.030	443.240	2 000 006	3 342 010 026	0	339 001 017	weblogic.work.ExecuteThread.run()
0.	439.557	0	3 272 009 816	0	334 001 002	weblogic.t3.srvr.SubsystemRequest.run()
0.	436.595	0	3 249 009 747	0	330 000 990	weblogic.application.utils.StateMachineDriver.nextState(weblogi
0.	410.677	0	3 085 009 255	0	312 000 936	weblogic.management.deploy.internal.ConfiguredDeployments.trans
0.010	403.883	0	3 047 009 141	0	311 000 933	weblogic.management.deploy.internal.ConfiguredDeployments.deplo
0.010	399.960	0	3 029 009 087	0	309 000 927	weblogic.management.deploy.internal.DeploymentServerService.res
0.	395.977	0	3 000 009 000	0	307 000 921	weblogic.management.deploy.internal.DeploymentServerService.sta
0.	326.899	0	3 302 009 906	0	302 000 906	Compile::Code_Gen()
0.	284.269	0	2 170 006 510	0	209 000 627	weblogic.security.acl.internal.AuthenticatedSubject.doAs(weblog
0.	283.859	0	2 167 006 501	0	209 000 627	weblogic.security.service.SecurityManager.runAs(weblogic.securi
0.	262.043	0	2 138 006 414	0	191 000 573	weblogic.application.internal.BaseDeployment$2.next(java.lang.O
0.	259.201	0	2 118 006 354	0	191 000 573	weblogic.application.internal.BaseDeployment.activate(weblogic.
0.	255.869	0	2 097 006 291	0	189 000 567	weblogic.application.internal.DeploymentStateChecker.activate(w

Two main paths: HotSpot compiler and weblogic
(Inferred from function names)

SOA: Filter by Function in Stack

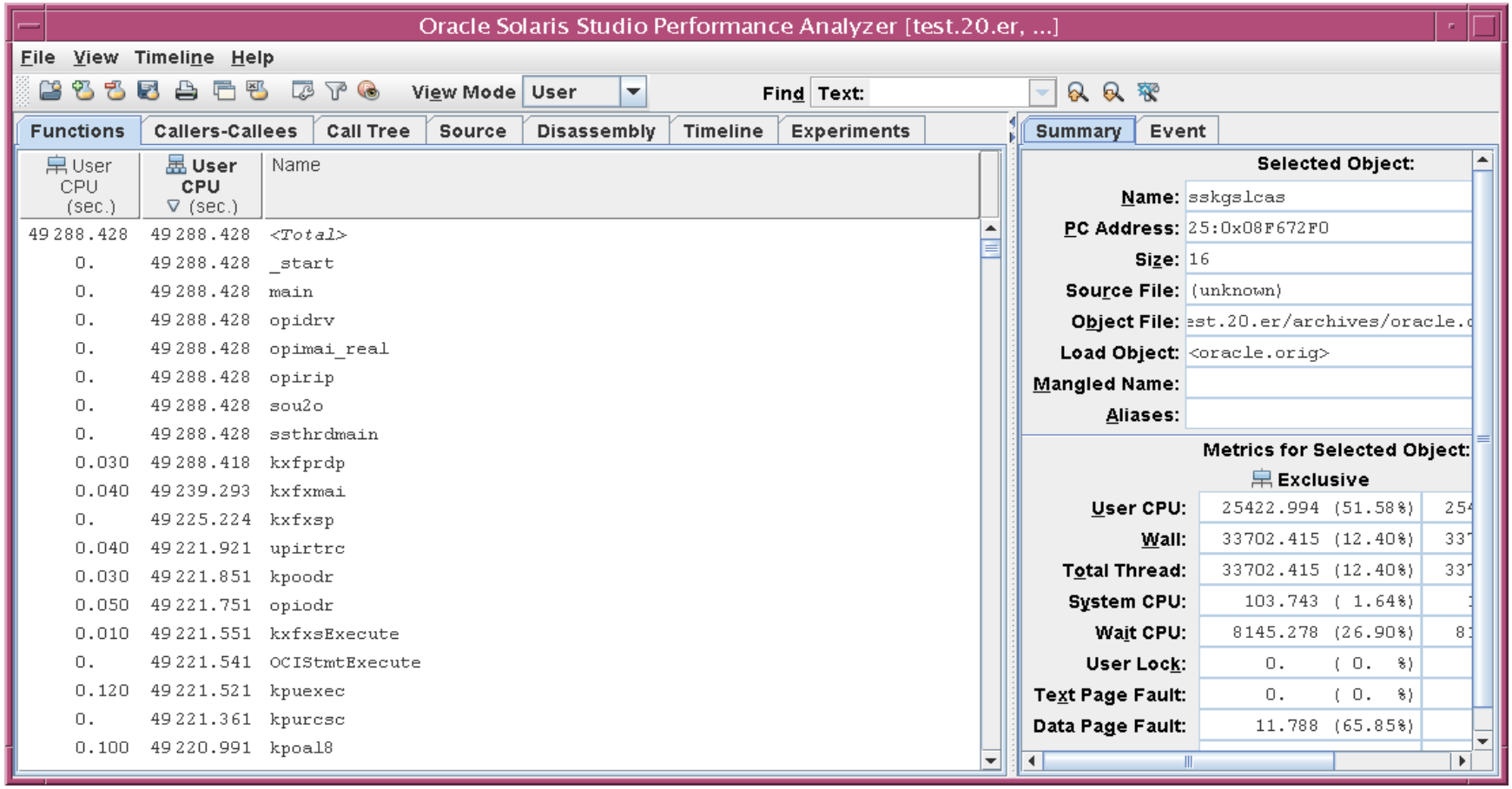


Function list shows data only from events with stacks containing `weblogic.work.ExecuteThread.execute()`

Oracle Database Profile

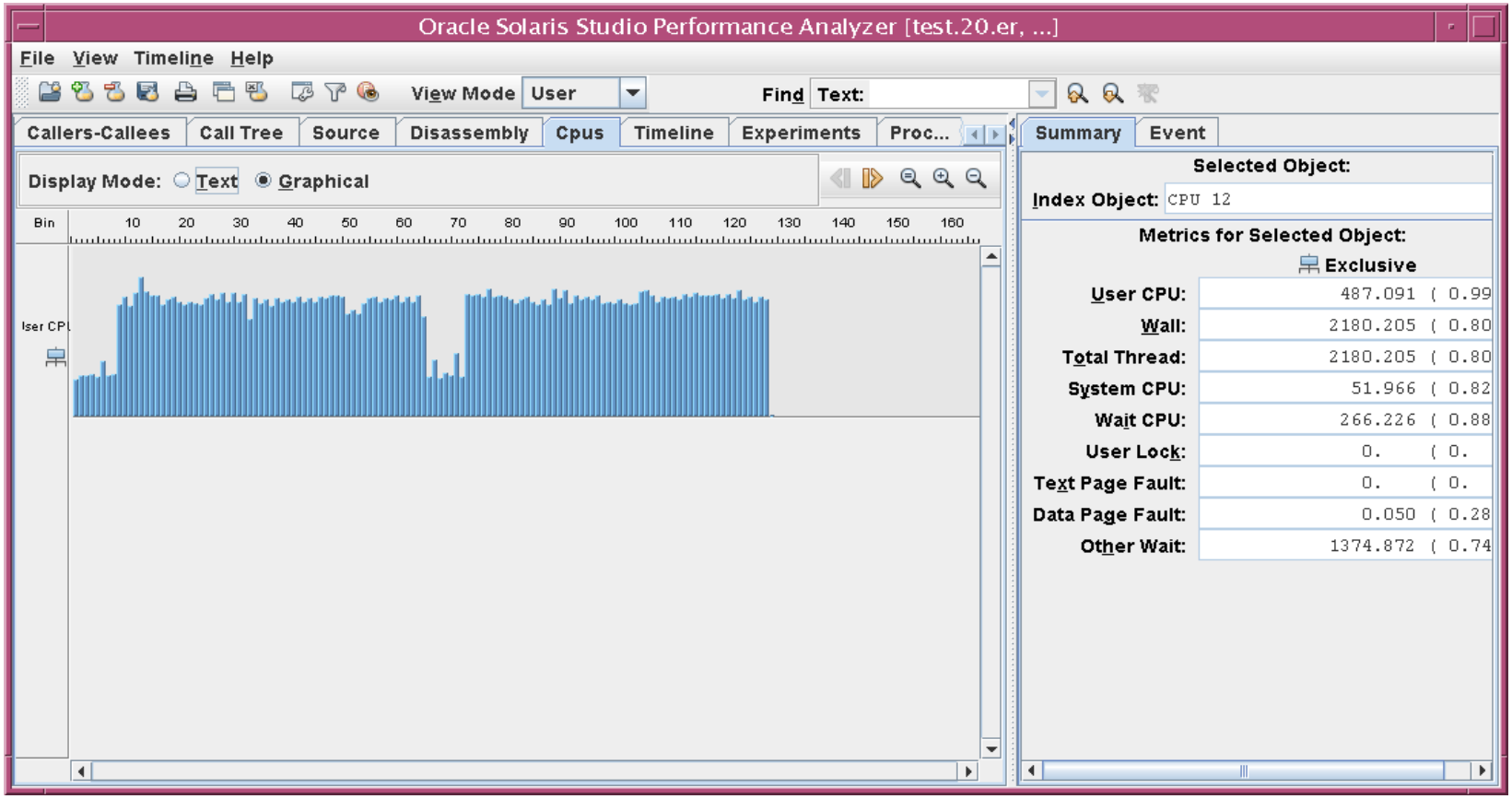
- Collected during TPC-H power test
- Script launches server, with **`-y USR`** flag
- Queries launched by a second script
  - Send **`SIGUSR`** to enable data collection
  - Run one query
  - Send **`SIGUSR`** to disable data collection
- Experiment has markers for each query
- Run on 128-CPU machine, uses 906 processes
  - Many are ephemeral, with no profile ticks
  - 256 processes do significant work

Oracle Database: Function List



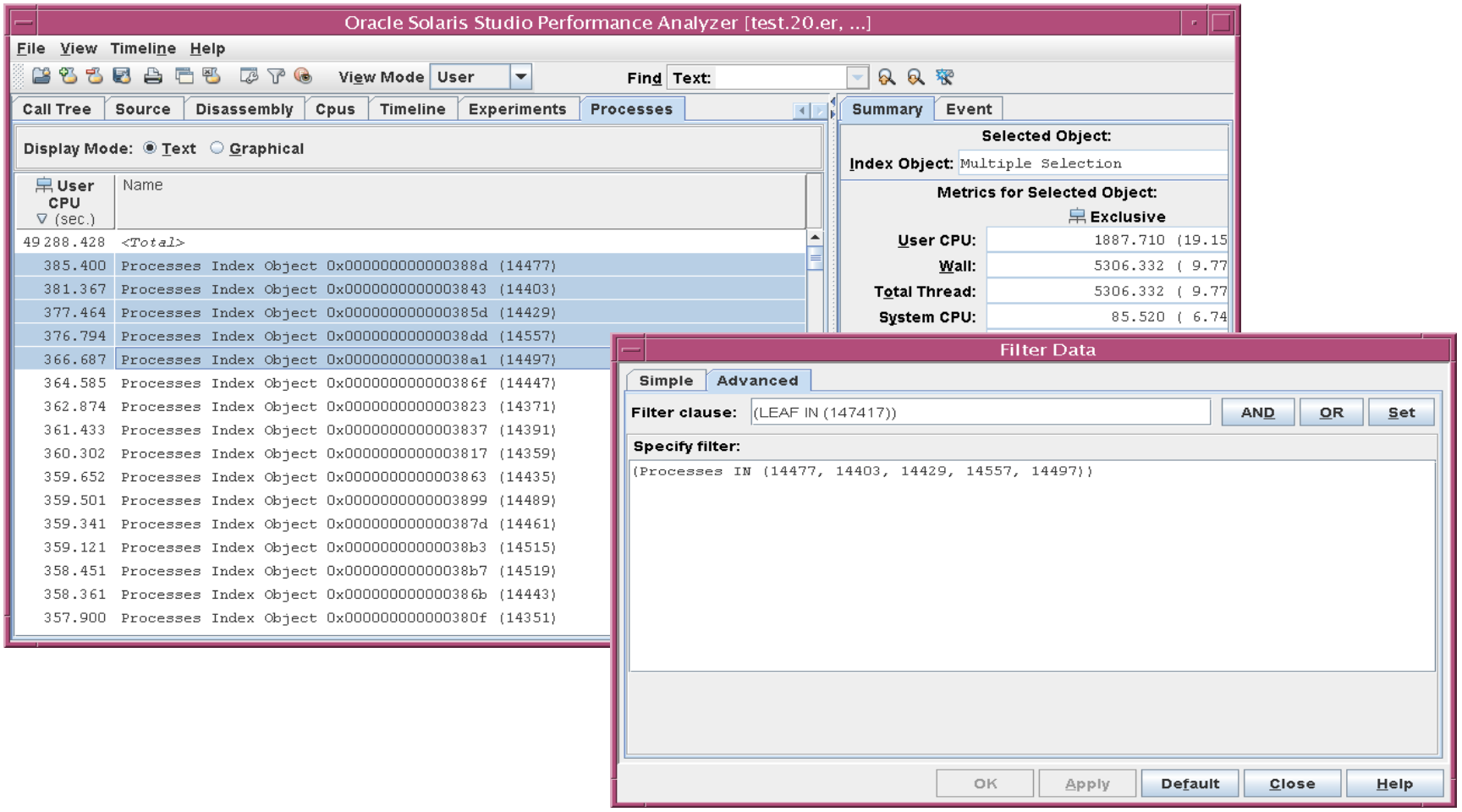
~40 minute run

Oracle Database: per-CPU Profile



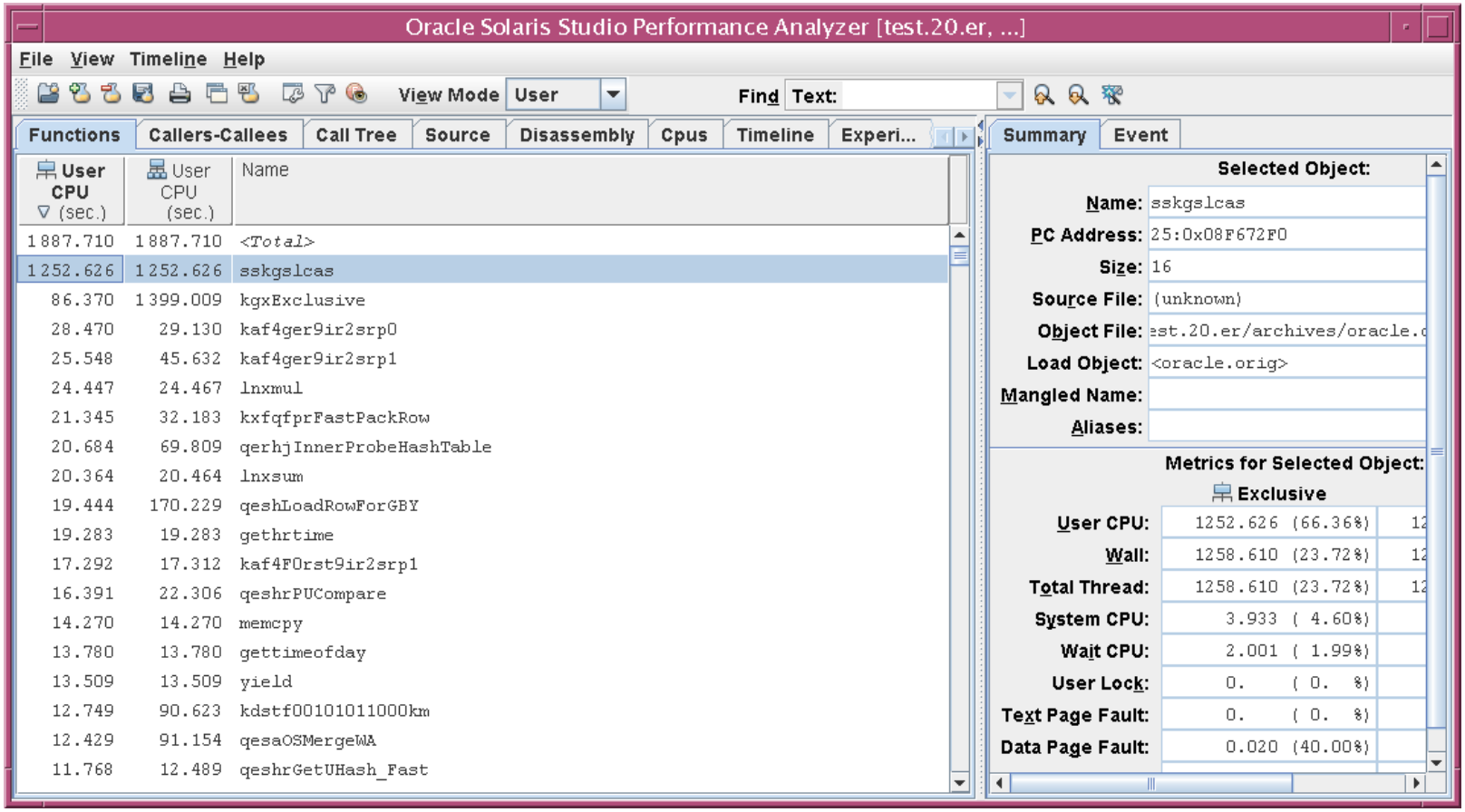
Sorted by CPU Number

Oracle Database: per-Process Profile



Per-process profile; filter set for top 5 processes

Oracle Database: Top Five Processes



Function list data filtered to show only the top 5 processes

SOFTWARE. HARDWARE. COMPLETE.


ORACLE®