Capacity Planning for Java Application Performance

Kingsum Chow
Nov 2-5, 2015

Agenda

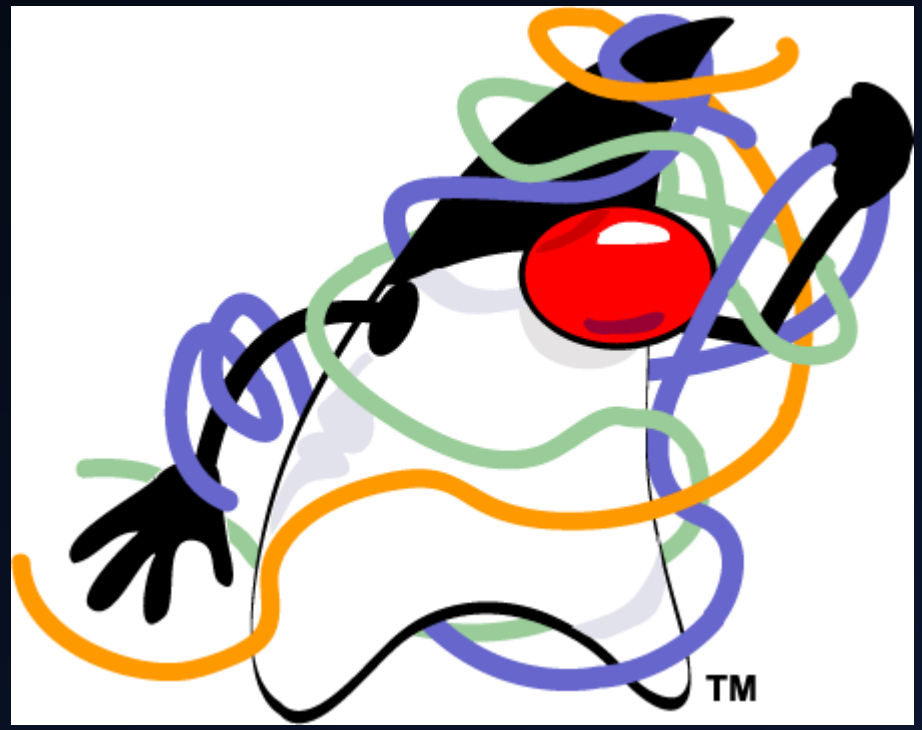
- Capacity Planning for Java Application Performance
- Overview of Java Virtual Machines
- Java Performance Monitoring
- Case Studies
- Summary

Background

- Java applications form an important class of applications running in the data center and the cloud. After a quick introduction to Java virtual machines, the characteristics of Java workloads will be described. It is then followed by general performance data collection and Java specific performance counters. A quick case study of the CPU-memory trade-offs of Java programs will be demonstrated using analytics. The knowledge can aid capacity planning for Java applications.

Capacity Planning

- The process of determining the amount of capacity required to produce in the future. This process may be performed at an aggregate or product-line level (resource requirements planning), at the master-scheduling level (rough-cut capacity planning), and at the material requirements planning level (capacity requirements planning).
- http://scm.ncsu.edu/scm-articles/scm-terms
- Throughput
- Response Times



Capacity Planning Process

Performance Data Collection

Data Cleansing Preparation

Model Construction Analytics

Capacity Planning

Overview of Java Virtual Machines

JIT	Instruction Selection Instruction Ordering Dynamic Recompilation Prefetch
GC	Cache Management Memory Layout Page Management
Classlibs	Threading Performance Tuning
Interpreter	Instruction Selection

Java Performance Monitoring

- User experience monitoring (e.g. faban driver)
- Platform monitoring (e.g. Java logs, gc logs)
- System monitoring (e.g. sar)
- CPU monitoring (e.g, perf)

SPEC Benchmarks

- The Standard Performance Evaluation Corporation (SPEC) is a non-profit corporation formed to establish, maintain and endorse a standardized set of relevant benchmarks that can be applied to the newest generation of high-performance computers. SPEC develops benchmark suites and also reviews and publishes submitted results from our member organizations and other benchmark licensees.

- https://www.spec.org/

SPECjvm2008

- SPECjvm2008 (Java Virtual Machine Benchmark) is a benchmark suite for measuring the performance of a Java Runtime Environment (JRE), containing several real life applications and benchmarks focusing on core java functionality.
- The suite focuses on the performance of the JRE executing a single application; it reflects the performance of the hardware processor and memory subsystem, but has low dependence on file I/O and includes no network I/O across machines.
- SPEC also finds user experience of Java important, and the suite therefore includes startup benchmarks and has a required run category called base, which must be run without any tuning of the JVM to improve the out of the box performance.
  - Leverages real life applications (like derby, sunflow, and javac) and area-focused benchmarks (like xml, serialization, crypto, and scimark).
  - Also measures the performance of the operating system and hardware in the context of executing the JRE.
- https://www.spec.org/jvm2008/

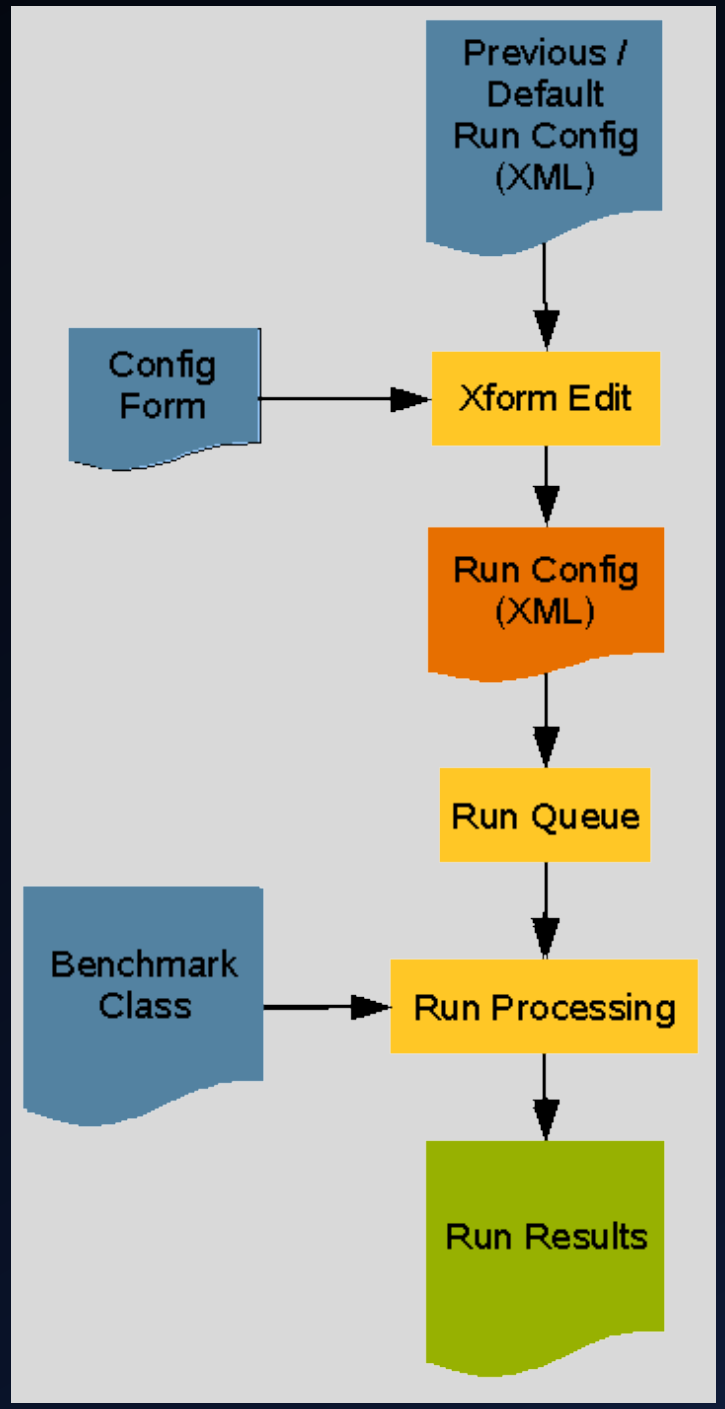
SPECjEnterprise2010

- The SPECjEnterprise2010 benchmark is a full system benchmark which allows performance measurement and characterization of Java EE 5.0 servers and supporting infrastructure such as JVM, Database, CPU, disk and servers.
- The workload consists of an end to end web based order processing domain, an RMI and Web Services driven manufacturing domain and a supply chain model utilizing document based Web Services. The application is a collection of Java classes, Java Servlets, Java Server Pages , Enterprise Java Beans, Java Persistence Entities (pojo's) and Message Driven Beans.
- SPECjEnterprise2010 is the third generation of the SPEC organization's J2EE end-to-end industry standard benchmark application. The new SPECjEnterprise2010 benchmark has been re-designed and developed to cover the Java EE 5.0 specification's significantly expanded and simplified programming model, highlighting the major features used by developers in the industry today. This provides a real world workload driving the Application Server's implementation of the Java EE specification to its maximum potential and allowing maximum stressing of the underlying hardware and software systems.
- https://www.spec.org/jEnterprise2010/

Faban Load Driver

- The harness uses the current (or default if this is the first run) run configuration file and the submission form to generate the web form for the run.
- Using the user-entered values, it then generates a new run configuration file for this run.
- The run is then placed on the run-queue to be executed by the run daemon.

- http://faban.org/1.2/docs/guide/harnessdev/overview.html



Other SPEC Java benchmarks

- SPEC JMS® 2007
  - The SPEC JMS® 2007 benchmark is the first industry-standard benchmark for evaluating the performance of enterprise message-oriented middleware servers based on JMS (Java Message Service). It provides a standard workload and performance metrics for competitive product comparisons, as well as a framework for in-depth performance analysis of enterprise messaging platforms.
- SPECjbb2005 (RETIRED: October 2013)
  - SPECjbb2005 (Java Server Benchmark) is SPEC's benchmark for evaluating the performance of server side Java. Like its predecessor, SPECjbb2000, SPECjbb2005 evaluates the performance of server side Java by emulating a three-tier client/server system (with emphasis on the middle tier). The benchmark exercises the implementations of the JVM (Java Virtual Machine), JIT (Just-In-Time) compiler, garbage collection, threads and some aspects of the operating system. It also measures the performance of CPUs, caches, memory hierarchy and the scalability of shared memory processors (SMPs).
- SPECjbb®2013 (2015 coming soon)
  - Defects Identified in SPECjbb®2013

Java Platform Monitoring

- Performance Impact of Garbage Collection
- **Throughput** focuses on maximizing the amount of work by an application in a specific period of time. Examples of how throughput might be measured include:
  - The number of transactions completed in a given time.
  - The number of jobs that a batch program can complete in an hour.
  - The number of database queries that can be completed in an hour.
- **Responsiveness** refers to how quickly an application or system responds with a requested piece of data. Examples include:
  - How quickly a desktop UI responds to an event
  - How fast a website returns a page
  - How fast a database query is returned
- http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html

System Activity Monitoring

- In computing, sar (System Activity Report) is a Unix System V-derived system monitor command used to report on various system loads, including CPU activity, memory/paging, device load, network. Linux distributions provide sar through the sysstat package.
- https://en.wikipedia.org/wiki/Sar_(Unix)

CPU monitoring

- perf began as a tool for using the performance counters subsystem in Linux, and has had various enhancements to add tracing capabilities.
- Performance counters are CPU hardware registers that count hardware events such as instructions executed, cache-misses suffered, or branches mispredicted. They form a basis for profiling applications to trace dynamic control flow and identify hotspots. perf provides rich generalized abstractions over hardware specific capabilities. Among others, it provides per task, per CPU and per-workload counters, sampling on top of these and source code event annotation.
- https://perf.wiki.kernel.org/index.php/Main_Page

Case Studies

- 1. Throughput
- 2. Response Time

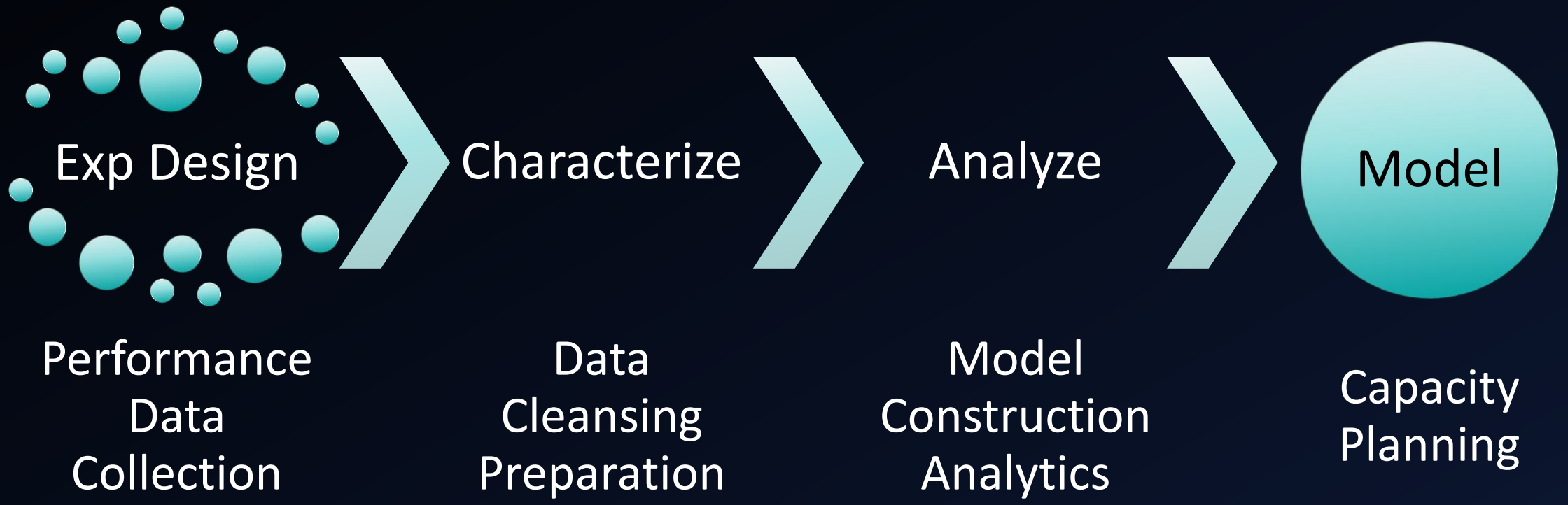
Case 1. Evaluating impact of key hardware features

- Throughput of Enterprise Java application depends on both hardware and software. We try to keep software configuration as similar as possible to quantify the impact of different hardware features
- With DOE we look at the impact of each individual feature and also the interaction between different features. This enables us to build a mathematical model that will estimate the throughput for the enterprise java application

Throughput Performance Model

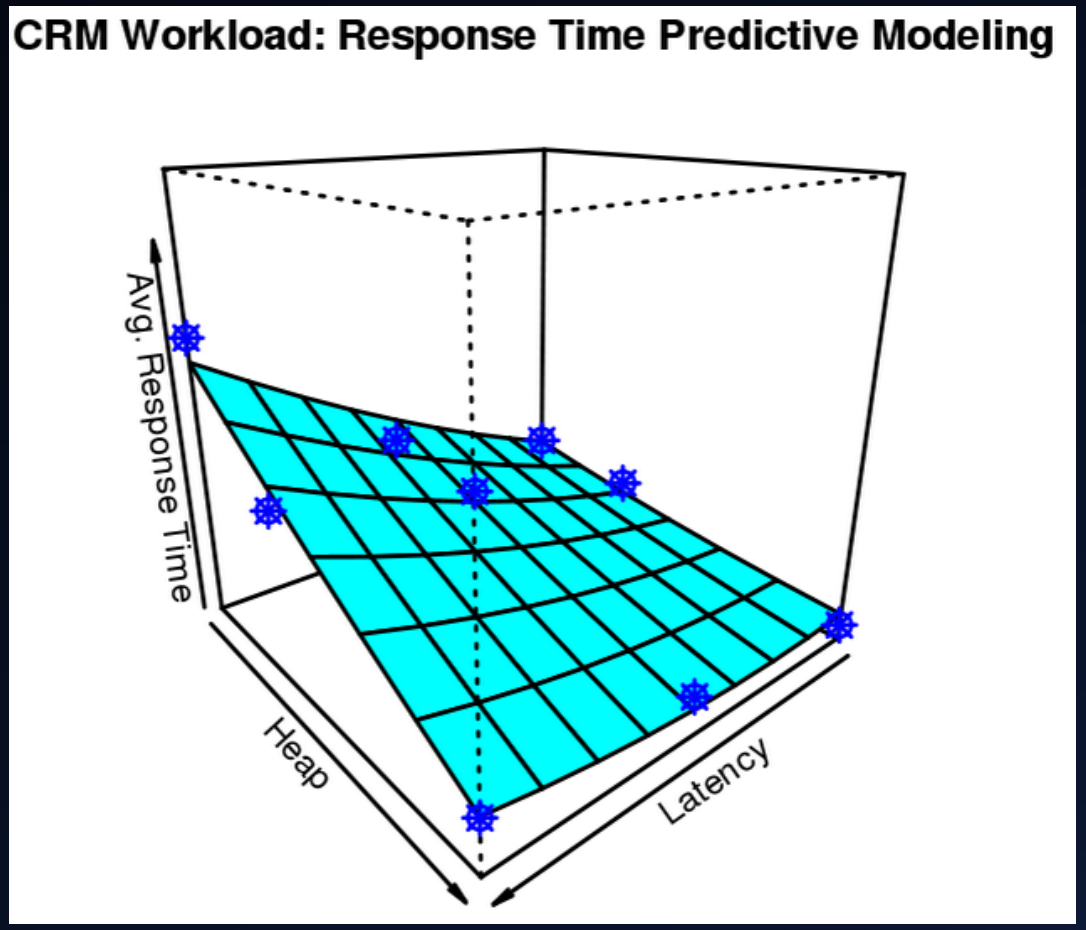
- Full Factorial Design for 4 hardware features
- Data collection
- Data Preparation
- Model Construction
- Assess Model Quality

Demo 1. Impact of Multiple Hardware Features



Case 2. Heap Memory Trade offs

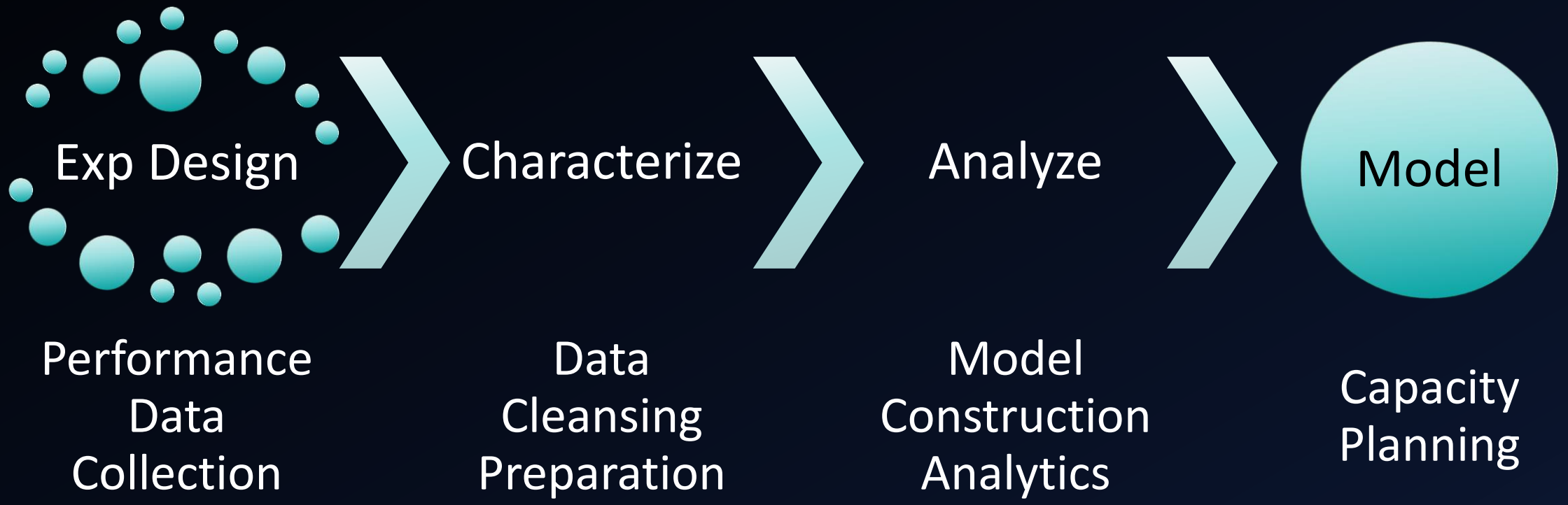
- Build a model
  - Response Time = f(Heap, Memory Latency)



Response Time Performance Model

- Full Factorial Design (Heap, Memory Speed)
- Data collection
- Data Preparation
- Model Construction
- Assess Model Quality

Demo 2. Heap Memory Trade-offs



Summary

- Characterize Java Applications through Performance Monitoring
- Identify user needs by throughput or response time requirements
- Distribute Resources for Capacity Planning