哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

Chapter 8: Software Construction for Performance

8.1 Performance Metrics for Software Construction

软件构造性能指标

Wang Zhongjie
rainy@hit.edu.cn

May 12, 2019

Outline

- **Performance Metrics 性能度量指标**
- **What you will learn in this chapter 本章学习内容**

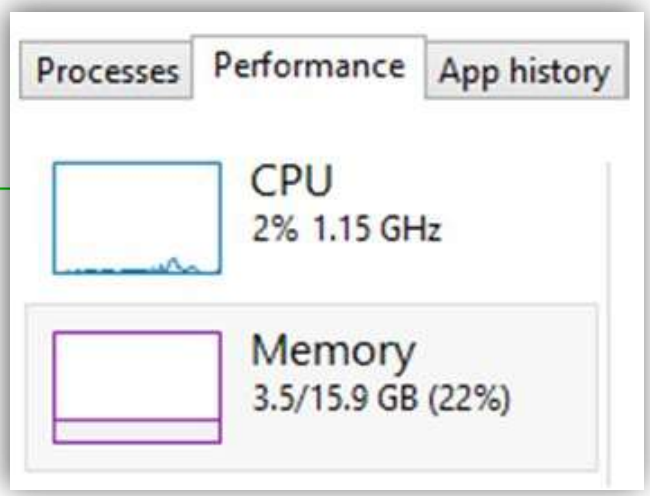
本章是课程覆盖的第5个质量指标：时空性能

这是大家最熟悉的指标，虽然很重要，但并非软件构造中最重要的指标，当其他指标得以优化之后，再去考虑性能问题。

Reading

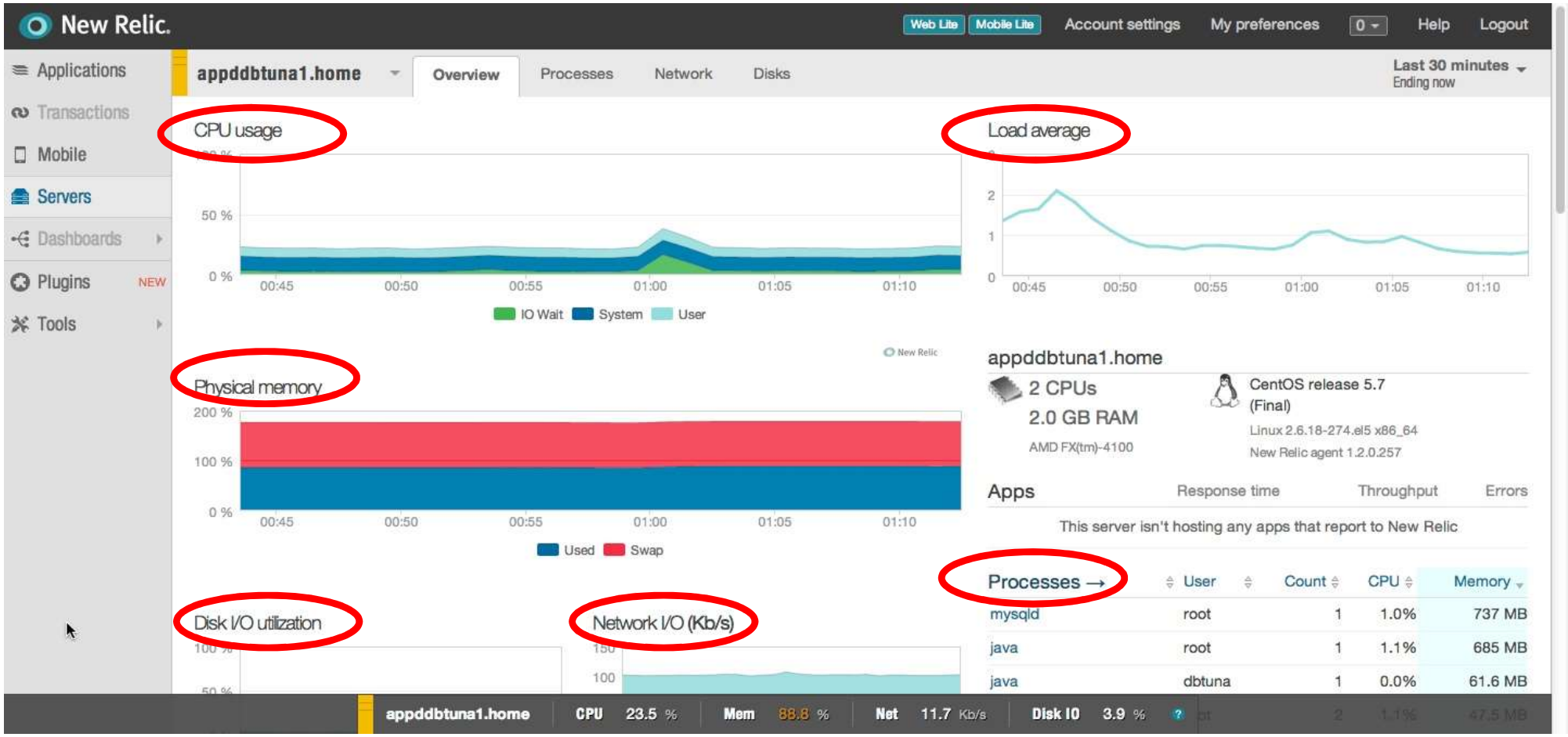
- **CMU 15-214：17**
- **Java性能权威指南：第1、2、4、5、6章**
- **Java编程思想：第13章、第18章**
- **Java Performance Tuning：第3章**

Performance

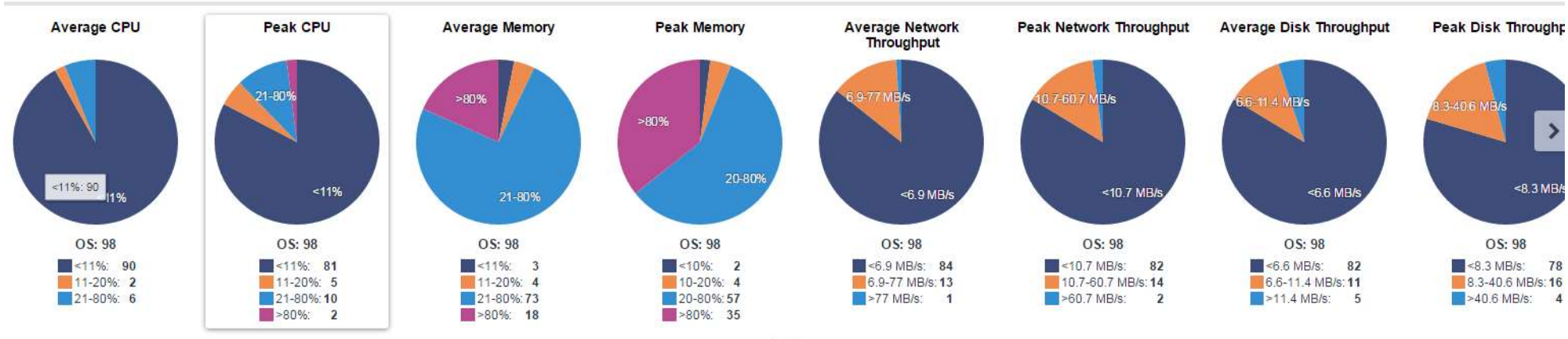


- **Performance is the amount of work accomplished by a computer system.**
  - Short response time for a given piece of work
  - High throughput (rate of processing work)
  - Low utilization of computing resources such as CPU, memory, disk
  - High availability of the computing system or application
  - Fast (or highly compact) data compression and decompression
  - High bandwidth
  - Short data transmission time
  - …
- **Performance of any computer system can be evaluated in measurable, technical terms, using one or more of the metrics.**
  - 有专门的课程讲解计算机系统（软件+硬件）的性能评价

Performance metrics



Performance dashboard



OS Performance (Peak CPU)

VIEW FULL REPORT EXPORT AS .CSV

Displaying 98 entries with 56 hidden columns

Filter by...

Name	Logical CPUs	Virtual CPUs	Avg CPU (24 hours)	Peak CPU (24 hours)	Avg CPU (7 days)	Peak CPU (7 days)	Avg CPU (30 days)	Peak CPU (30 days)
gpeqadb01	7	7	75.38%	93.79%	72.99%	99.89%	62.86%	99.94%
gvicvc6app01	2	2	37.25%	83.94%	62.25%	97.65%	55.53%	97.67%
gpeprddb01	24	24	30.49%	75.83%	26.82%	85.52%	26.32%	85.52%
gpedevdb01	24	24	43.74%	70.69%	48.19%	70.80%	32.44%	96.96%
ftp	2	2	2.03%	55.16%	2.12%	58.78%	1.85%	58.78%
nosgpe185	24	24	32.66%	54.73%	31.10%	63.77%	24.58%	63.77%
nosgpe184	48	24	23.43%	40.61%	23.24%	69.85%	22.49%	69.85%
gvicvio07	8	2	19.70%	32.60%	21.36%	32.60%	9.90%	32.60%
nosgpe181	24	12	8.45%	31.44%	22.63%	76.21%	32.60%	79.40%

Runtime program performance

- **Time performance** 时间性能
  - **Execution time** for a single statement, for a complex structure (such as a loop, I/O, etc), for the whole program; 每条指令、每个控制结构、整个程序的执行时间
  - **Distribution of execution time** of different statements/structures (relative time distribution) 不同语句或控制结构执行时间的分布情况
  - **Time bottleneck** of a program's execution 时间瓶颈在哪里？
- **Space performance** 空间性能
  - **Memory consumption** for a single variable, for a complex data structure, for the whole program; 每个变量、每个复杂结构、整个程序的内存消耗
  - **Distribution of memory consumption** of different variables/data structures (relative space distribution) 不同变量/数据结构的相对消耗
  - **Space bottleneck** of a program's execution 空间瓶颈在哪里？
  - **Evolution of memory consumption** along with time 随时间的变化情况

Factors influencing runtime performance

- **Space performance**
  - Algorithms
  - Data structure
  - Memory allocation
  - Garbage collection
- **Time performance**
  - Basic statements
  - Algorithms
  - Data structure
  - I/O (file, database, network communication, etc)
  - Concurrency / multi-thread / lock
- **In general, an operation is considered as being expensive if this operation has long runtime or high memory consumption.**

A average CPU can do approximately 1 billion (10^9) operations per second.

How to get memory consumption?

- **The total used / free memory of an program can be obtained in the program via `java.lang.Runtime.getRuntime()`, and the runtime has several method which relates to the memory.**

```
private static final long MEGABYTE = 1024L * 1024L;
public static long bytesToMegabytes(long bytes) {
    return bytes / MEGABYTE;
}

List<Person> list = new ArrayList<Person>();
for (int i = 0; i <= 100000; i++)
    list.add(new Person("Jim", "Knopf"));
// Get the Java runtime
Runtime runtime = Runtime.getRuntime();
// Run the garbage collector
runtime.gc();
// Calculate the used memory
long memory = runtime.totalMemory() - runtime.freeMemory();
System.out.println("Used memory is bytes: " + memory);
System.out.println("Used memory is megabytes: "
            + bytesToMegabytes(memory));
```

How to get execution time?

- **Use `System.currentTimeMillis()` to get the start time and the end time and calculate the difference.**

```
class TimeTest {
    public static void main(String[] args) {
        long startTime = System.currentTimeMillis();
        long total = 0;
        for (int i = 0; i < 10000000; i++) {
            total += i;
        }

        long stopTime = System.currentTimeMillis();
        long elapsedTime = stopTime - startTime;

        System.out.println(elapsedTime);
    }
}
```

java.lang.OutOfMemoryError

Class OutOfMemoryError

java.lang.Object
  java.lang.Throwable
    java.lang.Error
      java.lang.VirtualMachineError
        java.lang.OutOfMemoryError

All Implemented Interfaces:

Serializable

```
public class OutOfMemoryError
extends VirtualMachineError
```

Thrown when the Java Virtual Machine cannot allocate an object because it is out of memory, and no more memory could be made available by the garbage collector. `OutOfMemoryError` objects may be constructed by the virtual machine as if suppression were disabled and/or the stack trace was not writable.

```
$java -Xmx32m ErrorExample
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
        at ErrorExample.main(JavaOutOfMemoryErrorExample.java:5)
```

java.lang.StackOverflowError

Class StackOverflowError

java.lang.Object
    java.lang.Throwable
        java.lang.Error
            java.lang.VirtualMachineError
                java.lang.StackOverflowError

All Implemented Interfaces:

Serializable

```
public class StackOverflowError
extends VirtualMachineError
```

Thrown when a stack overflow occurs because an application recurses too deeply.

Too slow programs

What you will learn in this chapter

Contents in this chapter

- Runtime memory management 程序运行时的内存管理方法
- Memory performance and garbage collection 存储性能和垃圾回收
- I/O performance 输入/输出性能
- Algorithm performance 算法性能 (时空复杂性) --- 其他课程
- Dynamic performance analysis methods and tools 程序动态性能分析方法与工具
- Code tuning for performance optimization 面向性能优化的代码调优
- Design patterns for performance optimization 面向性能优化的设计模式

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

The end

May 12, 2019