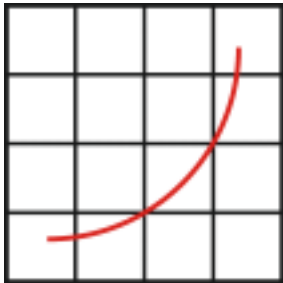


# SPEC – Power and Performance



**spec**<sup>®</sup>

Design Overview

SPECpower\_ssj2008 V1.11

Standard Performance Evaluation Corporation

## Table of Contents

<b>1 Introduction</b> .....	<b>3</b>
<b>2 The SUT and its Workload</b> .....	<b>3</b>
<b>3 Power and Temperature Measurement</b> .....	<b>3</b>
<b>4 Control and Collection</b> .....	<b>4</b>
<b>5 Configuration</b> .....	<b>4</b>
<b>6 Component version Numbering</b> .....	<b>5</b>
<b>7 Further Details</b> .....	<b>6</b>

## 1 Introduction

SPECpower\_ssj2008 is a benchmark product developed by the Standard Performance Evaluation Corporation (SPEC), a non-profit group of computer vendors, system integrators, universities, research organizations, publishers, and consultants. It is designed to provide a view of a server system's power consumption running Java server applications.

The general approach is to compare measured performance with measured power consumption. An initial requirement, as well, was to include power measurement data of a system running at different target load levels, to reflect the fact that datacenter server systems run at different target loads relative to maximum throughput.

SPEC also considers the ambient temperature during the benchmark measurement relevant to the results, and so temperature measurement is a requirement as part of a full benchmark report.

**To check for possible updates to this document, please see**  
[http://www.spec.org/power/docs/SPECpower\\_ssj2008-Design\\_Overview.pdf](http://www.spec.org/power/docs/SPECpower_ssj2008-Design_Overview.pdf).

## 2 The SUT and its Workload

For this benchmark product, the System Under Test (SUT) is a single-address-space server system or a homogeneous collection of such systems, sharing resources that are electrically relevant. The target loads are measured in terms of the performance of a fixed chosen workload that runs on the SUT. The workload is a Java application. It generates and completes a mix of transactions, and the throughput is the number of transactions completed per second over a fixed period. The application begins by running a calibration phase designed to determine the maximum throughput of the system, by generating transactions at the full rate that they can be completed. The calibration phase consists of at least three four-minute iterations, giving the Java runtimes the opportunity to apply just-in-time compilation and other runtime adaptations adequately; the maximum throughput used to calculate the target loads is the average of the throughput realized in the last two iterations in the calibration phase.

After the maximum throughput has been determined, the application then calculates the throughput values that correspond to each target load (100%, 90%, ... 20%, 10%, 0% of maximum as calibrated). After this is determined, the benchmark then enters the measurement interval during which the workload iterates through ten target loads, at each load generating transactions for completion at the target rate, using a randomized distribution to determine the times at which transactions are issued. By default, each load level lasts for 240 seconds (not including the ramp up and ramp down periods separating the measurement intervals, which will be covered later).

The 0% target load is also referred to as "active idle". In this case, the system is ready to accept transactions, but none are being issued.

The Java workload may be run in a single JVM instance, or multiple instances; in either case there will be another Java instance running the JVM Director, an application that controls and coordinates one or more JVMs running the workload.

For further details on the workload running on the SUT, see the SPECpower\_ssj2008 design document at [http://www.spec.org/power/docs/SPECpower\\_ssj2008-Design\\_ssj.pdf](http://www.spec.org/power/docs/SPECpower_ssj2008-Design_ssj.pdf)

## 3 Power and Temperature Measurement

As the workload described above runs, power measurements are recorded for later correlation with the performance at each target load, and for calculation of the benchmark metric. Temperature is also recorded to ensure that the ambient temperature during the run is no less than 20 degrees Centigrade.

Power is measured using one or more power analyzers, which sit between the wall A/C power and the SUT. In principle it would be possible for a user to gather the necessary data to characterize power efficiency with just the SUT, the benchmark workload, and a power analyzer.

There are technical requirements for the acceptance of a Power Analyzer to be used in running the benchmark compliantly (see the SPECpower\_ssj2008 Run Rules). Accepted analyzers support an automated data-gathering programming interface.

Temperature is measured using one or more temperature sensors. There are also requirements on the temperature sensors for acceptance to be used in a compliant benchmark run (see the SPECpower\_ssj2008 Run Rules). Accepted sensors support an automated data-gathering programming interface.

To simplify the process of gathering power and temperature data and synchronizing it with the performance data, and to make the process less error-prone, the benchmark in fact consists of the workload application on the SUT plus additional modules that are used to gather data and analyze the results.

The first of these is the SPEC Power Temperature Daemon (SPEC PTDaemon). This code runs on a Windows, Linux, or Solaris system connected to a power analyzer or a temperature sensor by a standard hardware interface, which could be serial, USB, or GPIB, depending on the available hardware and drivers. It communicates with the analyzer or sensor via the programming interface to the latter. PTDaemon itself is controlled via a simple socket interface, responding to requests for power (and temperature) readings. This interface is detailed in SPEC's Power and Temperature Daemon documentation. Multiple instances of the Power Temperature Daemon can be run, in cases where there are multiple power analyzers or temperature sensors.

PTDaemon supports an initial collection of power analyzers and temperature sensors, but the detailed design document referred to above explains how to extend support to another accepted one, and it is anticipated that the collection of supported analyzers will grow with new versions as they are released by SPEC.

## 4 Control and Collection

A second module is the Control and Collection System (CCS), which collects information from the workload on the SUT, and power and temperature information from the power and temperature daemons. It communicates with both through TCP/IP connections. The CCS is required to run on a system separate from the SUT, called the controller. The CCS is used to connect to three types of data sources via TCP/IP socket communication: the director JVM controlling the workload on the SUT, an instance of the PTDaemon connected to a power analyzer, and an instance of the PTDaemon connected to a temperature sensor.

For details on the CCS, see the CCS Design Document at [http://www.spec.org/power/docs/SPECpower\\_ssj2008-Design\\_ccs.pdf](http://www.spec.org/power/docs/SPECpower_ssj2008-Design_ccs.pdf)

## 5 Configuration

The most basic SPECpower\_ssj2008 test bed implementation that can generate a compliant benchmark run consists of at least four devices:

- a server under test (SUT), with some number of JVMs running the workload
- a power analyzer, controlled by an instance of PTDaemon
- a temperature sensor, controlled by another instance of PTDaemon
- a controller system, with a JVM running CCS
- (optionally) a director system, with a JVM running the director. The director JVM can also be run on the SUT or the controller system

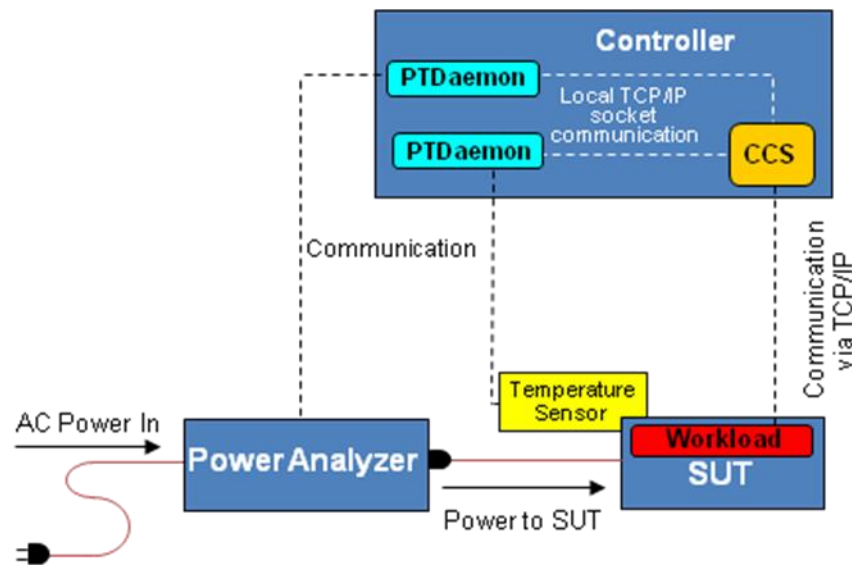
Within a given test bed implementation, there are usually several ways to distribute and run the various software modules of the SPECpower\_ssj2008 benchmark suite.

Each of the components above could conceivably run on its own hardware. All components could run on one system. By following the User's Guide and using the properties in the properties files, one could distribute the work in a variety of ways. The benchmark package is designed to allow a maximum of flexibility to enable the benchmark user to analyze the benchmark's behavior to the broadest extent.

The key restriction for a compliant run is that the CCS should run on a separate system from the SUT. The purpose of this restriction is to guarantee that the network communication between the SUT and CCS is remote.

By editing properties files, it is also possible to modify many other behaviors of the benchmark. The User's Guide provides details on how one might change the order of running of different target loads, and a collection of other options that allow deep analysis of the benchmark.

The simplest possible SPECpower\_ssj2008 test bed implementation is illustrated in the figure below. Here we assume that the JVM Director is running on the SUT with Workload.



As options beyond this picture, the JVM director could run on any of these systems or even another one, with appropriate configuration and property file changes. The PTDaemon components controlling the power analyzer and temperature sensor could run on the SUT or even a separate system from the benchmark controller system.

Beginning with version 1.10 of SPECpower\_ssj2008, submissions are permitted that incorporate a SUT including multiple nodes, each a single address space system, multiple power analyzers, and multiple temperature sensors.

Further and more detailed information about CCS can be found in the CCS Design Document at [http://www.spec.org/power/docs/SPECpower\\_ssj2008-Design\\_ccs.pdf](http://www.spec.org/power/docs/SPECpower_ssj2008-Design_ccs.pdf)

## 6 Component version Numbering

The current benchmark packaging consists of loosely coupled components, and the goal is that they be usable in multiple benchmarks. As a result multiple versions of these components may be delivered in different releases of a single benchmark or across multiple different benchmarks.

Furthermore, one benchmark may require different levels of each component from another benchmark.

As part of managing these relationships we are supporting separate versioning of each component. The relationship of the benchmark versions to component version will be managed by SPEC.

Beginning with this release of SPECpower\_ssj2008, we are moving to a three-part numbering scheme for version numbers of the components CCS, PTDaemon, and ssj (the SUT workload). So for example, each of these will appear in a given benchmark kit with a version number of the form x.y.z.

Minor changes to a component that require no update to this new revision would be reflected by a change in the z component of the version number. An example would be a change in PTDaemon to support a new power analyzer; users need not move to this revision unless they explicitly require this support.

Changes that would require users to move to a new revision would be reflected in a change in the y component. If an existing power analyzer were changed to be non-compliant, the resulting change to PTDaemon would allow PTDaemon to continue to work with the other components, but would also be a change we expected to be reflected in kits used for submission of results. In a case like this, if PTDaemon were at level 2.3.2, the new version following the change would be 2.4.0.

An API-incompatible change, that causes a component no longer to work with the others, would be reflected by a change in the x component.

The initial change in any of the higher level version numbers also causes a reset of the lower-level ones to 0.

## 7 Further Details

Detailed information on how to run the benchmark, and how to generate compliant SPECpower\_ssj2008 submission data can be found in the component design documents mentioned above, and particularly in the User's Guide and Run and Reporting Rules for the benchmark.

### 7.1 Trademark

Product and service names mentioned herein may be the trademarks of their respective owners.