Standard Performance Evaluation Corporation (SPEC®)

The SERT® Suite Design Document 2.0.x

7001 Heritage Village Plaza, Suite 225
Gainesville, VA 20155,
USA

Table of Contents

This document version: https://www.spec.org/sert2/SERT-designdocument-20220223.pdf
Latest SERT 2.x.x version: https://www.spec.org/sert2/SERT-designdocument.pdf
Previous version: https://www.spec.org/sert2/SERT-designdocument-20210331.pdf

1. Introduction

The SERT suite was created by the Standard Performance Evaluation Corporation, the world's leading organization for benchmarking expertise. The SPECpower Committee designed, implemented, and delivered the SERT suite, a next-generation standard for measuring and evaluating the energy efficiency of servers. The SERT suite was created with the input from leaders of various global energy-efficiency programs and their stakeholders in order to accommodate for their regional program requirements.

1.1. About SPEC

SPEC was formed by the industry in 1988 to establish industry standards for measuring compute performance. SPEC has since become the largest and most influential benchmark consortium world-wide. Its mission is to ensure that the marketplace has a fair and useful set of metrics to analyze the newest generation of IT equipment.

SPEC has a long history of designing, developing, and releasing industry-standard computer system performance benchmarks in a range of industry segments, plus peer-reviewing the results of benchmark runs. Performance benchmarking and the necessary work to develop and release new benchmarks can lead to disagreements among participants. Therefore, SPEC has developed an operating philosophy and range of normative behaviors that encourage cooperation and fairness among diverse and competitive organizations:

- Decisions are reached by consensus. Motions require a qualified majority to carry.
- Decisions are based on reality. Experimental results carry more weight than opinions. Data and demonstration overrule assertion.
- Tools and benchmarks should be architecture-neutral and portable.

The SPEC community has developed more than 30 industry-standard benchmarks for system performance evaluation in a variety of application areas and has provided thousands of benchmark licenses to companies, resource centers, and educational institutions globally. Organizations using these benchmarks have published more than 50,000 peer-reviewed performance reports on SPEC's website (https://www.spec.org/results.html).

1.1.1. SPEC Membership

SPEC membership is open to any interested company or entity. The members and associates of SPEC's Open System Group (OSG) are entitled to licensed copies of all released OSG benchmarks and tools as well as unlimited publication of results on SPEC's public website. An initiation fee and annual fees are due for members. Nonprofit organizations and educational institutions have a reduced annual fee structure. Further details on membership information can be found on https://www.spec.org/osg/joining.html or requested at info@spec.org. Also, a current list of SPEC members can be found here: https://www.spec.org/spec/membership.html.

1.1. The SPECpower Committee

The increasing demand for energy-efficient IT equipment has resulted in the need for power and performance benchmarks. In response, the SPEC community established the SPECpower Committee, an initiative to augment existing or create new industry-standard benchmarks and tools with a power/energy measurement. Leading engineers and scientists, representing a diverse and competitive set of companies, in the fields of benchmark development and energy efficiency made a commitment to tackle this task. The development of the first industry-standard benchmark that measures the power and performance characteristics of server-class compute equipment began on January 26, 2006.

In December of 2007, SPECpower_ssj2008 was released, which exercises the CPUs, caches, memory hierarchy, and the scalability of shared memory processors on multiple load-levels. The benchmark runs on a wide variety of operating systems and hardware architectures. In version 1.10, which was released on April 15, 2009, SPEC augmented SPECpower_ssj2008 with multi-node support (e.g., blade-support). Several enhancements and code changes to all benchmark components, documentation updates, and run and reporting rules enhancements were included in version 1.11, released September 13, 2011 and version 1.12, released March 30, 2012.

The work on the SERT suite started around 2009 with the ground-breaking design of the Chauffeur™ framework followed with the design and implementation of the workload, automated hardware and software discovery, and GUI. After a series of successful Alpha and Beta test phase, the SERT suite was released on 26. February 2013. The U.S. Environmental Protection Agency (EPA) adopted the SERT suite for their ENERGY STAR for Server Certification program 2 month later on March 15, 2013. The SPECpower Committee released four upgrades over the years, in order to further enhance the usability of the SERT suite, and to increase its platform support

For version 2.0, one of the main focuses was the reduction of the SERT runtime and improvements on the automated hardware and software discovery, in order to save the user testing time and costs. The support of additional power analyzers and interfaces was adopted as well in order to provide a wider choice for users. Also, some major improvements of the memory subsystem worklets, for more accurate characterization of memory, were implemented. Simultaneously to these efforts, the SPECpower Committee in collaboration with the more academic-focused SPEC Power Research Group, conducted large scale experiments in order to determine the server efficiency metric for SERT 2.0 which enabled its release on March 14, 2017.

1.2. The SERT 2.0 Suite Overview

The SERT suite is providing a first order of approximation of server efficiency across a broad range of application environments. It is designed to be economical and easier to use by utilizing a comprehensive **graphical user interface** (GUI) and predetermined sets of **tuning parameters**. The SERT suite has been implemented and tested for a variety of **64-bit processors, operating systems**, and **JVMs**. It is scalable and tested up to a maximum of 8 **processor sockets** and a maximum of 64 **nodes**. The server under test (SUT) may be a single stand-alone server or a multi-node set of servers. A **multi-node** SUT will consist of server nodes that cannot run independently of shared infrastructure such as a backplane, power-supplies, fans, or other elements. These shared infrastructure systems are commonly known as “blade servers” or “multi-node servers”. Only identical (homogenous) servers are supported in a multi-node SUT configuration.

The main body of code is in written in Java in order to lower the burden of cross-platform support. The framework is designed to accommodate other **program languages** as well. A series of different **workload** characteristics are implemented to demonstrate the effectiveness of different server sizes and configurations. The workload also utilized SPEC’s concept of multiple **load levels** and details can be found in section 7. Regulatory programs are designed to foster continuous improvement, with thresholds for success rising as the industry progresses. The SERT suite is designed to match this paradigm by implementing a quick adoption process of new computing technologies. This process is described in SERT’s Run and Reporting Rules and the current list of **supported platforms** can be located here: https://www.spec.org/sert2/SERT-JVM_Options-2.0.html.

The right balance between high repeatability of the results, high sub-system coverage, and low resource allocation is desirable and resulted in a **run time** of around 2 ½ hours. The results output, including the SERT metric, are generated in both machine- and human-readable forms, enabling automatic submission to government-sponsored certification programs as well as both summary and detail reports for use by potential customers.

1.2.1. Feedback and Support Mechanism

Since the beginning of the SERT Beta phase in September 2011, the technical support is being conducted by volunteers from our member institutions. In general, the user sent a request via the SERT Support Request Form (https://www.spec.org/sert/feedback/issuereport.html) or discusses an items via the SERT Discussion Forum (https://www.spec.org/forums/index.php). Feedback is addressed on a first-in, first-out bases and since the official release of the SERT in February 2013 all requests have been successfully addressed.

1.1. Trademark and Copyright Notice

2. The SERT Suite Architecture

2.1. Environment Overview

The SERT suite is composed of multiple software components and shares design philosophies and elements from SPECpower_ssj2008 in its overall architecture.

For the most basic SERT measurement setup the following is required:

- **System under Test (HW)** – the actual system for which the measurements are being taken. The controller and SUT are connected to each other via a TCP/IP connection.
- **Controller (HW, e.g. server, PC, laptop)** – the system to which the power and temperature sensor are connected. Multi-Controller environments are supported.
- **Power analyzer (HW)** – connected to the Controller and used to measure the power consumption of the SUT. Multiple Power Analyzer environments are supported.
- **Temperature sensor (HW)** – connected to the Controller and used to measure the ambient temperature where the SUT is located.
- **The SPEC PTDaemon (SW)** – connects to the power analyzer and temperature sensor and gathers their readings while the suite of workloads executes. All instances of the PTDaemon (each must be the same version) must run on the Controller, each listening on a different port.
- **The Reporter (SW)** – summarizes the environmental, power, and performance data after a run is complete and compiles it into an easy to read format.
- **The GUI (SW)** – eases configuring and executing the SERT suite.

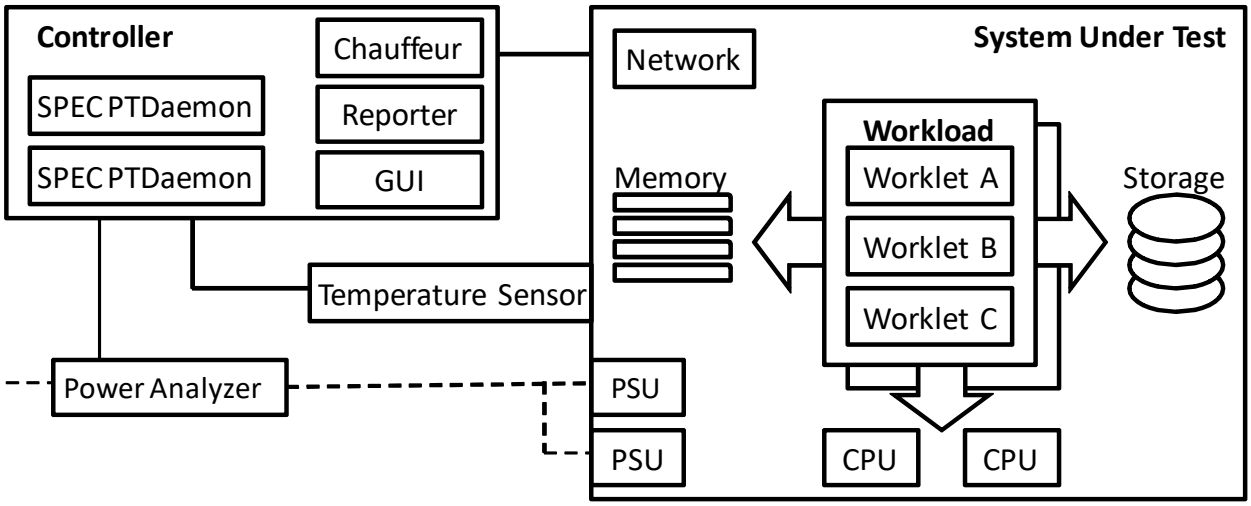


Figure 1: SERT System Diagram

Chauffeur (SERT test harness) handles the logistical side of measuring and recording power data along with controlling the software installed on the SUT and Controller. It is responsible for the initialization of the following JVMs:

- The Client JVM – executes the workload.
- The Host JVM – starts all Client JVMs on one SUT.
- The Director – instructs the Host JVM to start the workload.

The basic system overview in Figure 2: *SERT Components* shows these components in relationship to each other.

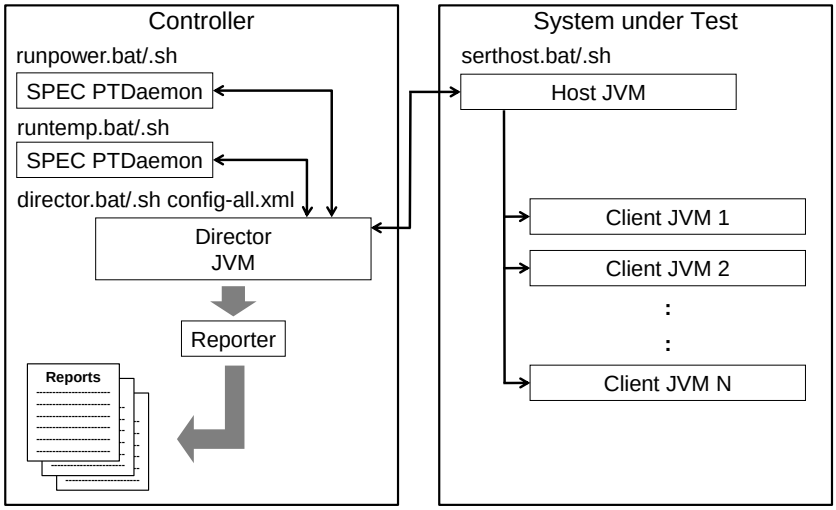


Figure 2: SERT Components

2.2. SERT Suite

The SERT Suite is composed of several logical elements including:

User

- A User is a representation of an external agent that can initiate work (e.g. a human being).
- Each User may maintain identifying information (e.g. each User represents a Warehouse).
- Each User may maintain temporary state information that persists from one transaction to another.

Worklet (Compress, CryptoAES, LU, SHA256, SOR, SORT, Flood3, Capacity3, SSJ, Idle)

- A Worklet defines a set of transactions that can be executed by a particular type of User.

Scenario

- A Scenario is a group of one or more Transactions which are executed in sequence by a particular User.
- When a worklet is running a load level, each scenario is scheduled to being execution at a specific time. If that time is in the future, the User will execute no transactions until the scheduled time arrives.
- Each transaction in a scenario is submitted to a JVM-wide thread pool for execution.

Interval

- Each Interval includes pre-measurement, recording, and post-measurement periods.
- Transactions are executed throughout the entire interval, but results are only recorded during the recording period.
- Power consumption is recorded only during the recording period.

Sequence (A sequence of related intervals)

- The intervals in a sequence may be executed identically (e.g. during calibration).
- The intervals in a sequence may execute at different load levels (e.g. 100%, 80%, 60%, 40%, and 20% of the calibrated throughput).
- The intervals in a sequence may run with different configuration parameters (e.g. floot3 runs Flood_Full and Flood_Half to utilize all of system memory and half of system memory, respectively).

Phase

- A phase of execution: warm-up, calibration, or measurement
- Each phase consists of a sequence of intervals.
- Chauffeur supports multiple sequences in the measurement phase, but SERT always runs a single sequence in each phase.

Workload (CPU, Memory, Storage, Idle)

- A workload is a group of worklets designed to stress some aspect of the SUT.
- The worklets in each workload run one at a time in a defined order.

2.3. Workload

The design goal for the SERT suite is to include all major aspects of server architecture, thus avoiding any preference for specific architectural features which might make a server look good under one workload and show disadvantages with another workload. The SERT workloads take advantage of different server capabilities by using various load patterns, which are intended to stress all major components of a server uniformly.

2.4. Worklet Execution Phases

The SERT test suite consists of several workloads which are designed to stress the various components of the system under test (SUT): CPU, memory, and storage. Each workload includes one or more worklets which execute specific code sequences to focus on one of these system components. The overall design structure of the SERT is shown in Figure 3: SERT Suite Components.

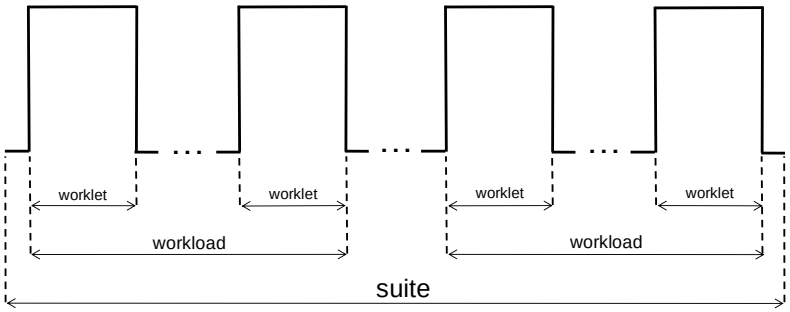


Figure 3: SERT Suite Components

Worklets are run one at a time. Most worklets consist of a warm-up phase, a calibration phase, and a measurement phase; some worklets do not include warm-up or calibration. Each of these phases consists of a sequence of one or more intervals. Each interval includes a pre-measurement, recording, and post-measurement period. Score calculations are based on the number of transactions and the power consumed during the recording period only. All of the worklets other than Flood3 use fixed time periods for the pre-measurement, recording, and post-measurement periods. Flood3 measures memory bandwidth while running a minimum number of iterations during each period. The pre-measurement period allows the worklet to reach steady state before recording begins. The pre- and post-measurement periods also ensure that in multi-client and multi-host runs, all clients are executing transactions concurrently during the recording period, even if there are slight discrepancies in the time that the recording period begins and ends. Intervals are separated by short delays to summarize results and allow new power analyzer ranges to take effect.

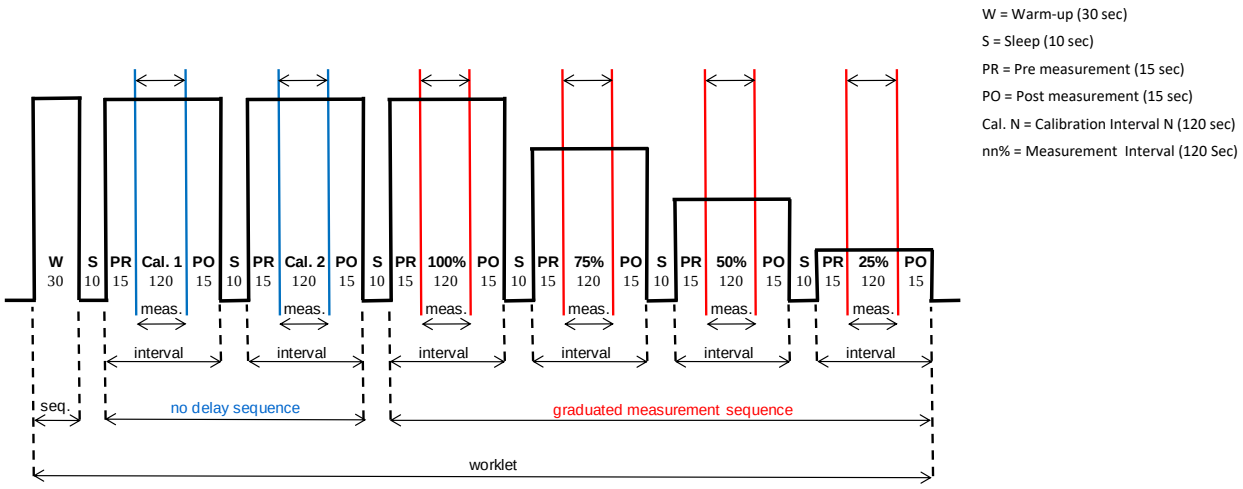


Figure 4: Graduated Measurement Execution

Most worklets (with the exception of the memory and idle worklets) run as shown in Figure 4: Graduated Measurement Execution. Each of these worklets begins with a warm-up phase which runs transactions for a short period of time to allow the worklet execution to stabilize. Throughput during the warm-up interval may vary due to factors such as JIT compilation, memory allocation, and garbage collection, and system caches adjusting to the new load. After warm-up, a sequence of calibration intervals is run to identify the maximum rate that transactions can be executed for this worklet. Two calibration intervals are used, and the calibrated throughput is the average for these two intervals. After the calibrated throughput is found, SERT runs a series of intervals where the worklet is throttled back to run at some percentage of the maximum throughput. The series of load levels varies by worklet, as defined in Section 7.

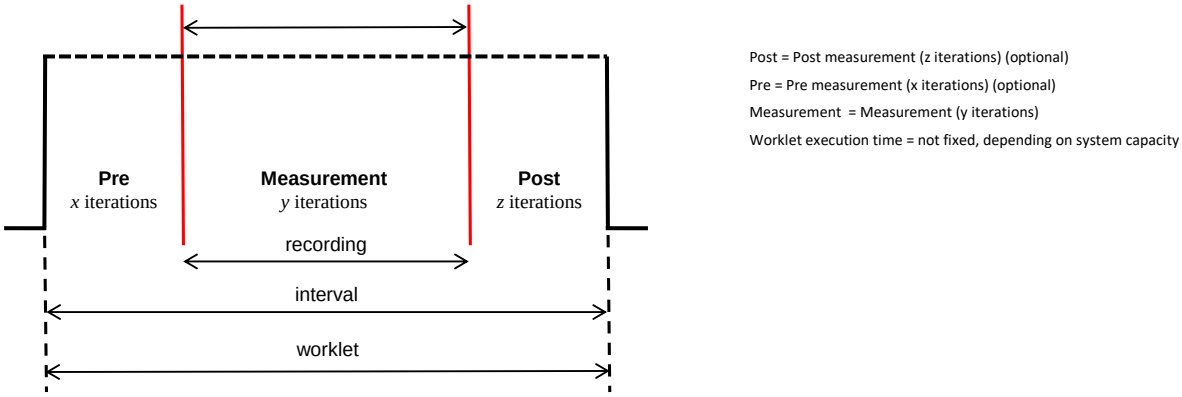


Figure 5: Fixed Iteration Execution

3. Power and Temperature Measurements

The SERT suite provides the ability to automatically gather measurement data from accepted power analyzers and temperature sensors and integrate that data into the SERT metric. The analyzers and sensors must be supported by the measurement framework, and must be compliant with the specifications in this section. The SERT suite is designed for environments that have a supply voltage tolerance of ± 5%.

3.1. Environmental Conditions

Power measurements need to be taken in an environment representative of the majority of usage environments. The intent is to discourage extreme environments that may artificially impact power consumption or performance of the server, before and during the SERT run.

The following environmental conditions need to be met:

- Ambient temperature lower limit: 20°C
- Ambient temperature upper limit: within documented operating specification of the SUT
- Elevation and Humidity: within documented operating specification of the SUT
- Overtly directing air flow in the vicinity of the measured equipment in a way that would be inconsistent with normal data center practices is not allowed.

3.2. Temperature Sensor Specifications

Temperature must be measured no more than 50mm in front of (upwind of) the main airflow inlet of the SUT. To ensure comparability and repeatability of temperature measurements, SPEC requires the following attributes for the temperature measurement device used during the SERT run:

- Logging - The sensor must have an interface that allows its measurements to be read by the SERT harness. The reading rate supported by the sensor must be at least four samples per minute.
- Accuracy - Measurements must be reported by the sensor with an overall accuracy of +/- 0.5 degrees Celsius or better for the ranges measured during the SERT run.

3.3. Power Analyzer Requirements

To ensure comparability and repeatability of power measurements, the following attributes for the power measurement device are required for the SERT. Please note that a power analyzer may meet these requirements when used in some power ranges, but not in others, due to the dynamic nature of power analyzer Accuracy and Crest Factor. The usage of power analyzer's auto-ranging function is not permitted.

- Measurements - The analyzer must report true RMS power (watts) and at least two of the following measurement units: voltage, amperes, and power factor.
- Accuracy - Measurements must be reported by the analyzer with an overall uncertainty of 1% or better for the ranges measured during the benchmark run. Overall uncertainty means the sum of all specified analyzer uncertainties for the measurements made during the benchmark run.
- Calibration - The analyzer must be able to be calibrated by a standard traceable to NIST (U.S.A.) (http://nist.gov) or a counterpart national metrology institute in other countries. The analyzer must have been calibrated within the past year.
- Crest Factor - The analyzer must provide a current crest factor of a minimum value of 3. For analyzers which do not specify the crest factor, the analyzer must be capable of measuring an amperage spike of at least three times the maximum amperage measured during any one-second sample of the benchmark run.
- Logging - The analyzer must have an interface that allows its measurements to be read by the SPEC PTDaemon. The reading rate supported by the analyzer must be at least one set of measurements per second, where set is defined as watts and at least two of the following readings: voltage, amperes, and power factor. The data

averaging interval of the analyzer must be either one (preferred) or two times the reading interval. "Data averaging interval" is defined as the time period over which all samples captured by the high-speed sampling electronics of the analyzer are averaged to provide the measurement set.

Examples:

An analyzer with a vendor-specified accuracy of +/- 0.5% of reading +/- 4 digits, used in a test with a maximum power value of 200W, would have "overall" accuracy of (((0.5%*200W)+0.4W)=1.4W/200W) or 0.7% at 200W.

An analyzer with a wattage range 20-400W, with a vendor-specified accuracy of +/- 0.25% of range +/- 4 digits, used in a test with a maximum power value of 200W, would have "overall" accuracy of (((0.25%*400W)+0.4W)=1.4W/200W) or 0.7% at 200W.

3.4. SPEC PTDaemon®

SPEC (Power & Temperature Daemon) PTDaemon is used by the SERT to offload the work of controlling a power analyzer or temperature sensor during measurement intervals to a system other than the SUT. It hides the details of different power analyzer interface protocols and behaviors from the SERT software, presenting a common TCP-IP-based interface that can be readily integrated into different benchmark harnesses.

The SERT harness connects to PTDaemon by opening a TCP port and using the simple commands. For larger configurations, multiple IP/port combinations can be used to control multiple devices.

PTDaemon can connect to multiple analyzer and sensor types, via protocols and interfaces specific to each device type. The device type is specified by a parameter passed locally on the command line on initial invocation of the daemon.

The communication protocol between the SERT and PTDaemon does not change regardless of device type. This allows the SERT to be developed independently of the device types to be supported.

3.5. Supported and Compliant Devices

The SERT suite utilizes SPEC's accepted measurement devices list and SPEC PTDaemon update process. See Device List (https://www.spec.org/power_ssj2008/docs/device-list.html) for a list of currently supported (by the SPEC PTDaemon) and compliant (in specifications) power analyzers and temperature sensors.

The process to add software support for a power analyzer or temperature sensor to the infrastructure can be found on the Power Analyzer Acceptance Process page (https://www.spec.org/power/docs/SPEC-Power_Analyzer_Acceptance_Process.pdf).

3.6. Power Analyzer Setup

The power analyzer must be located between the AC Line Voltage Source and the SUT. No other active components are allowed between the AC Line Voltage Source and the SUT. Power analyzer configuration settings that are set by the SPEC PTDaemon must not be manually overridden.

3.6.1. 3-Phase Measurements

A 3-phase analyzer must be used for 3-phase measurements.

3.7. DC Line Voltage

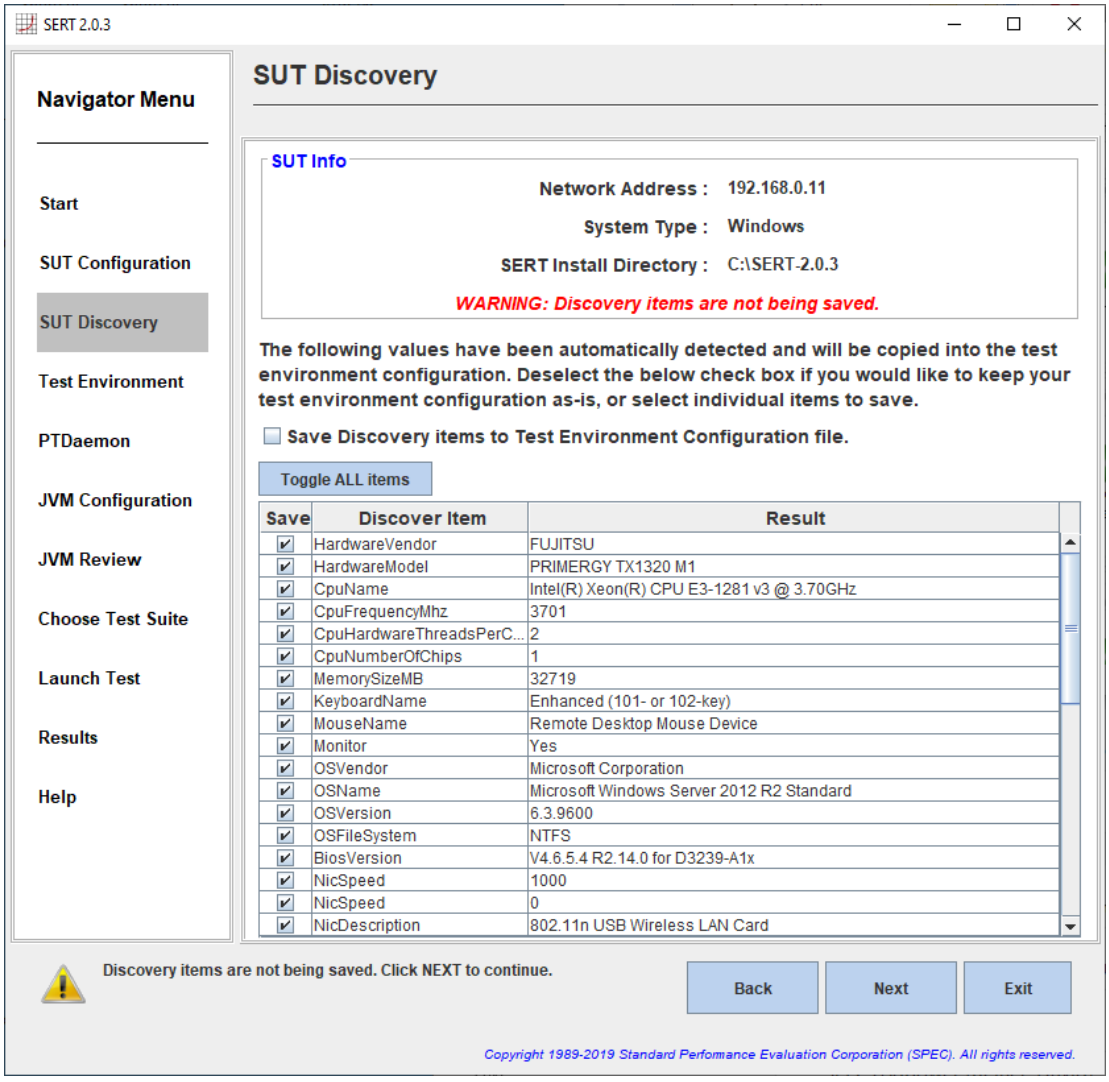
The SPECpower Committee is in favor of including DC support if new resources from companies whose focus is DC computing become available to the SPECpower Committee to address the development and support opportunity.

4. Graphical User Interface

A graphical user interface (GUI) is provided to facilitate configuration and setup of test runs, to allow real-time monitoring of test runs, and to review the results. The SERT GUI leads the user through the steps of configuring the SUT and initiating a test sequence. These steps include detecting and customizing the hardware and software configurations, selecting the test type, and displaying the results.

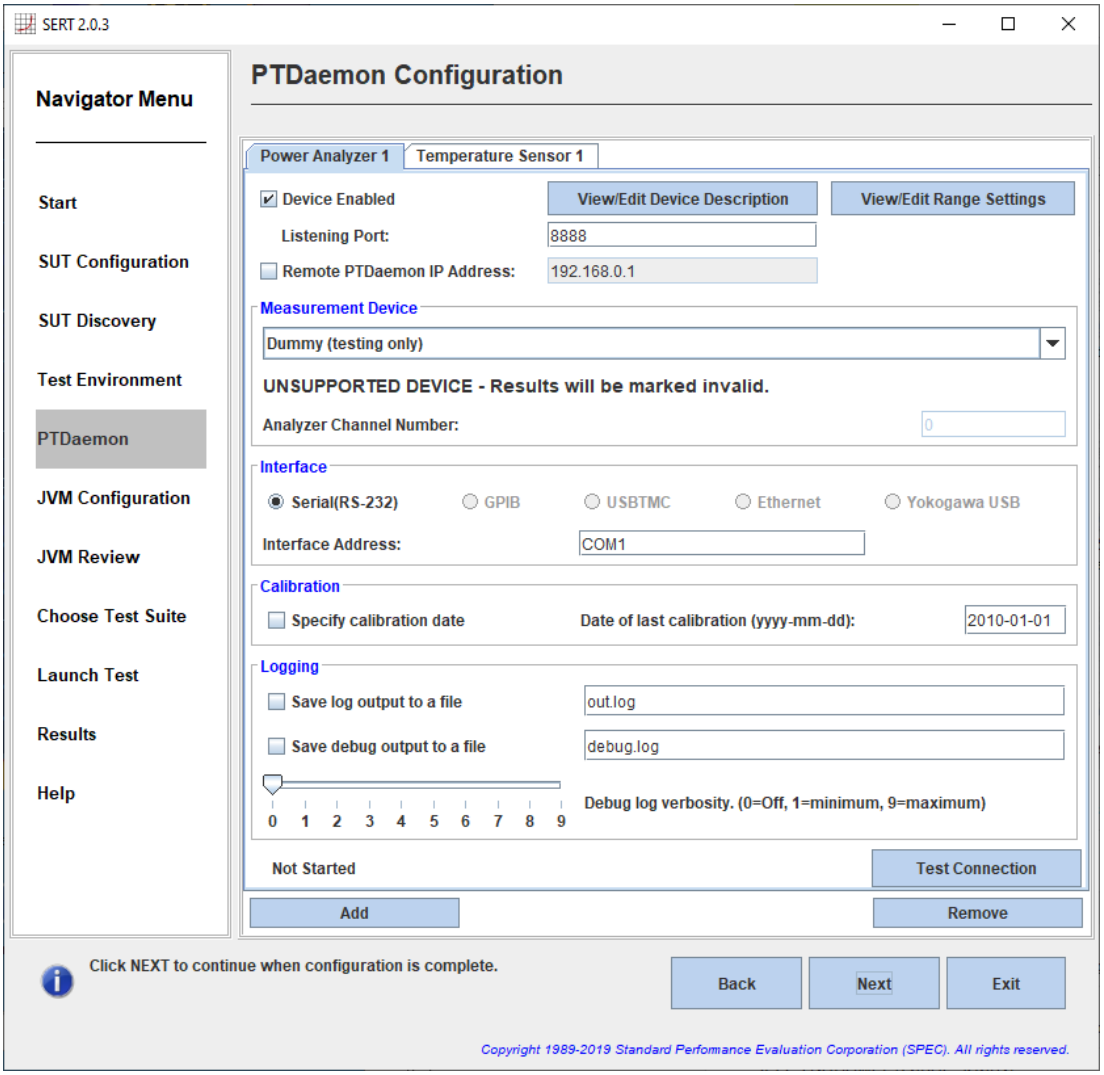
The SERT GUI includes several features to enable SERT testing with minimal training, thereby enhancing the accuracy of results. Some of the features include:

- Easy Navigation with Tabbed Screens via “Navigation Menu” and/or “Back/Next” buttons
- In-line usage guidance and help
- SUT Configuration Discovery (Detect function) automatically detects configuration information and populates many fields with SUT and Controller hardware and software details.

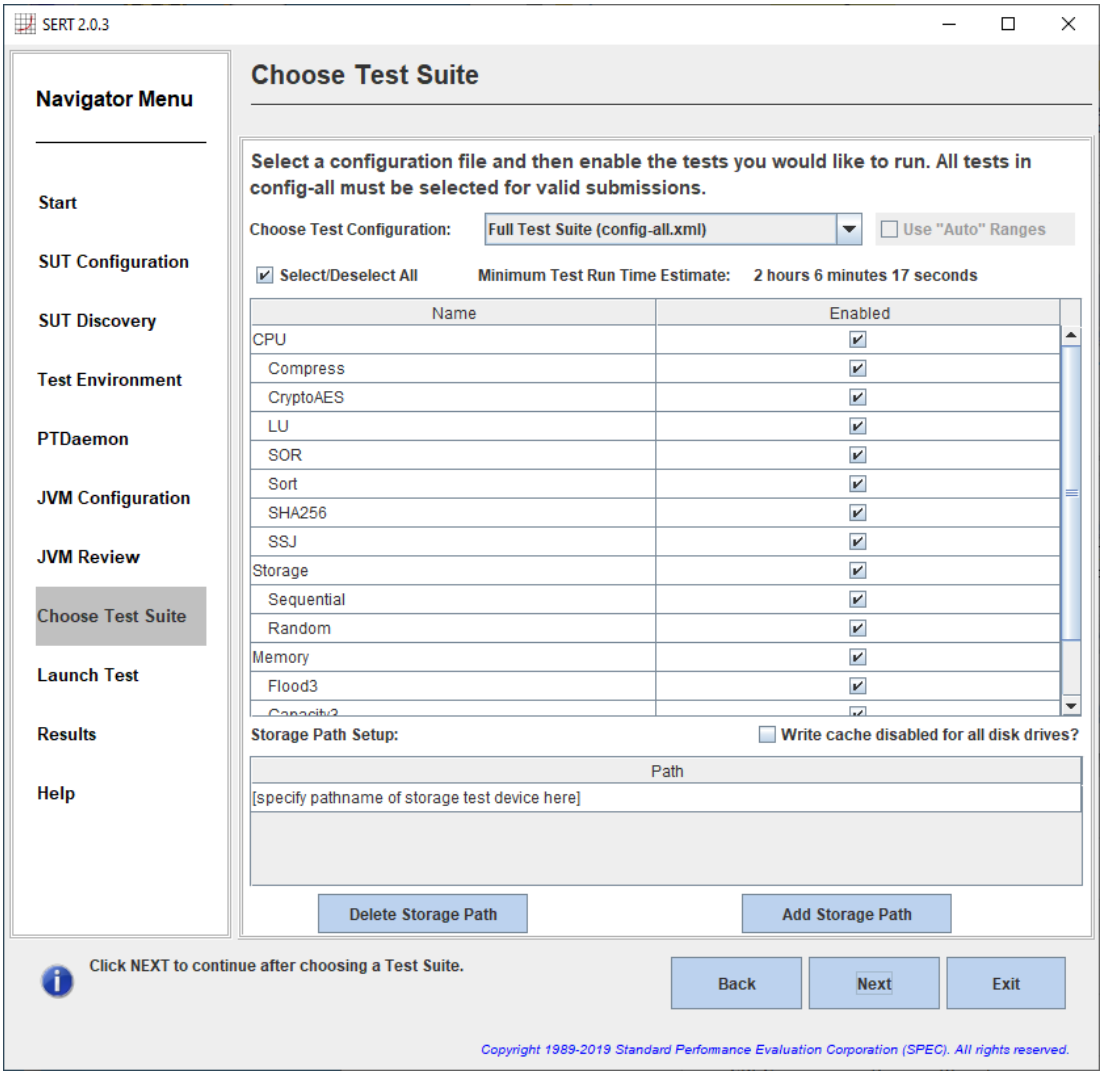


- The GUI displays test environment information, allowing the modification and saving of updated fields.
  - For use in reports: e.g. Vendor Information, Platform Configuration, Run-Time parameters, etc.

  - Test Run information files can be stored for subsequent reuse, thus saving time when testing multiple identical platforms. For running on multiple platforms that are similar to each other, the saved information for one platform can be loaded and quickly modified for the other platform.
- The GUI allows setting up the PTDaemon Configuration, and testing of the connection to a Power Analyzer or Temperature Sensor (as shown on the tabs in the screenshot below).
  - Support for all PTDaemon supported Power Analyzers and Temperature Sensors
  - Access to all PTDaemon debugging and configuration options
  - Establish and test a connection with PTDaemon supported devices
  - Progress, warnings, and errors are displayed in Launch Test panel output



- The GUI provides the capability to set up the Test Suite and Configuration.
  - Selects entire test suite for an official test run
  - Selective execution of a subset of workloads and worklets for a quick validation of test setup or experimentation
  - Configures JVM options and number of clients
  - Sets up one or multiple storage path(s) pointing to the physical storage device(s)



- Displays the Test Execution and Progress
  - Validates storage path parameters and storage device space remaining
  - Start measurements
  - Displays progress, warnings, and errors
  - PTDaemon messages displayed in separate tabs for clarity
- Displays Results and provides access to the final report
  - Save, View, and Recall archived reports

5. SERT 2 Metric, Reports, Validation, and Logging

5.1. The SERT 2 Metric

All SERT worklets, except Idle run at multiple load levels. For each of those load levels, energy efficiency is calculated separately. We define per load level energy efficiency Eff_{load} as follows (Eq 1):

$$Eff_{load} = \frac{\text{Normalized Performance}}{\text{Power Consumption}} \quad (1)$$

Normalized performance is normalized throughput for most worklets, with memory worklets being a notable exception. Power consumption in this context is the average measured power consumption for each load level. As such, the per load level energy efficiency score represents the amount of (normalized) transactions that were executed per unit of energy (Joule).

The worklet efficiency score $Eff_{worklet}$ is calculated using the geometric mean of the separate load level scores (Eq. 2). The geometric mean has been chosen in favor of the arithmetic mean or sum. The major difference between the geometric mean and arithmetic mean is that the arithmetic mean favors load levels with higher efficiency scores. Traditionally, higher load levels also feature higher efficiency scores on most systems. As a result, the arithmetic mean usually favors the results of high load levels. A change in energy efficiency at a higher load level has a greater impact on the final score than it would at a lower load level. The geometric mean, on the other hand, treats relative changes in efficiency equally at each load level. Finally, the resulting geometric mean is multiplied with a factor of 1000. This is a cosmetic factor that has been chosen in order to move the resulting score into a number range that is easier to read for a human reader.

$$Eff_{worklet} = \exp\left(\frac{1}{n} \times \sum_{i=1}^{n} ln(Eff_{load_i})\right) \times 1000 \quad (2)$$

n represents the number of load levels per worklet and Eff_{load_i} the energy efficiency for load level i.

5.1.1. Workload Efficiency Calculation

Workload efficiency $Eff_{workload}$ is calculated by aggregating the efficiency scores of all worklets within the workload using the geometric mean (Eq. 3). The worklet efficiency scores being aggregated are the results of the score calculation in Eq. 2 in Section 5.1.

$$Eff_{workload} = \exp\left(\frac{1}{n} \times \sum_{i=1}^{n} ln(Eff_{worklet_i})\right) \quad (3)$$

n represents the number of worklets per workload and $Eff_{worklet\_i}$ is the energy efficiency for each specific worklet.

5.1.2. SERT 2 Metric Calculation

The SERT 2 metric (or SERT 2 Efficiency Score) is the final aggregate of the workload scores. It too, is derived using the geometric mean. In contrast to the other mean aggregates, the SERT 2 metric does not consider all workloads equally. Instead, it uses a weighted geometric mean, putting a different focus on each of the workload scores. The particular workload weights are as follows:

- CPU weight: 65% **(High)**
- Memory weight: 30% **(Medium)**
- Storage weight: 5% **(Low)**

The weights have been decided by expert groups and represent a realistic weighting of the importance of the different workloads when comparing SERT to real world operation workloads. Note, again, that the workloads do not purely exercise the hardware component after which they are named. The CPU workloads also exercise some memory, the Memory workloads perform some work on the CPU, etc. This means that the weighted CPU workloads

(with their weight of 65%) already include some memory power and memory energy efficiency components. The weights have been created with this in mind. Consequently, the SERT 2 metric is calculated as follows (Eq. 4):

$$\text{SERT 2 Efficiency Score} = \exp(0.65 * \ln(\text{Eff}_{\text{CPU}}) + 0.3 * \ln(\text{Eff}_{\text{Memory}}) + 0.05 * \ln(\text{Eff}_{\text{Storage}})) \quad (4)$$

For further details on impact of server configuration on the SERT metric, see https://www.spec.org/sert2/SERT-metric.pdf.

5.2. Reference Platform

In order to get performance values in the same order of magnitude from all worklets, the individual worklet performance scores are normalized relative to an arbitrarily selected baseline in the 2.0 release of the SERT.

The baseline results have been derived from 6 SERT runs (3 per software stack) on this reference platform:

- 1-socket rack server (1U)
- 1 x Intel(R) Xeon(R) CPU E3-1240 @ 3.30GHz
- 2 x 4GB 2Rx8 PC3 -10600E DIMMs
- 1 x SATA 160GB 7.2k rpm 3.5" HDD connected to an onboard SATA controller
- Software Stack 1: RedHat Enterprise Linux 6.7 with the Oracle Java SE Runtime Environment (build 1.7.0_80-b15)
- Software Stack 2: Microsoft Windows Server 2008 R2 Datacenter (6.1.7601) with Oracle Java SE Runtime Environment (build 1.7.0_80-b15)

The reference score for each worklet is defined as the average worklet performance score over all 6 SERT runs on the configurations described above.

5.3. Reporting and Output Files

The SERT produces multiple reports, and includes code that will ensure the authenticity of the reports.

5.3.1. Results Output

In order to reduce the effort of displaying and/or storing the desired information, the primary report output (result.xml) is generated in the XML format. This report output contains all the information regarding the SERT run including all hardware and software configuration of the Controller, the SUT, and the SERT workloads. It includes all pertinent information about the worklets like JVM affinity, options, and other launch settings, along with the resulting performance, power and efficiency results.

5.3.2. Report Formats

Four human readable versions of the report are also generated, two in the HTML format (.html) and two in plain text (.txt).

The "**results.html/.txt**" reports, contain all pertinent hardware and software information about the test environment, including the SUT. It also includes the workload efficiency scores, idle power consumption and the SERT efficiency metric. The "**results-details.html/.txt**" reports, contain all the information present in the results.html / .txt file above, along with detailed breakdown of the performance and power at each load level for all the worklets. In addition, for each worklet, the run-time parameters like number of CPUs, threads, and command line options are also captured.

These reports are designed to be viewed on a screen with minimum width of 1280 pixels and contain a subset of data that is derived from the primary XML output.

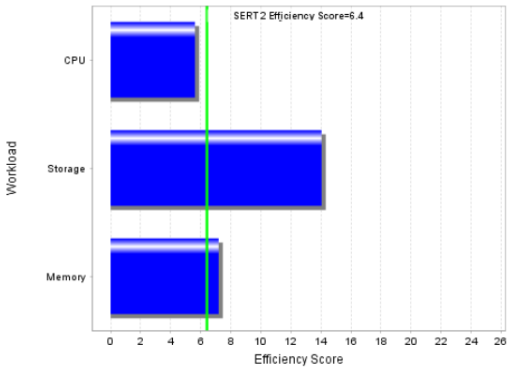
5.3.3. Sample Report Output

The top part of a **resuts.html** report from a sample system is shown in Figure 6. The left part of the **Summary** graphically displays the Workload Efficiency Scores for the three different workloads (CPU, Storage, and Memory) as well as the SERT 2 Efficiency Score (SERT 2 metric). An overview of the server configuration is shown on the right. The actual information shown here corresponds to the default information which comes with the SERT installation package. This is artificial information and doesn't describe a real system configuration. Figure 7 shows the same information in a plain text format.

SERT™ Report

Hewlett Packard Enterprise ProLiant DL120 G7 (Historical System)			
Test Sponsor	Hewlett Packard Enterprise	**Software Availability**	Apr-2016
Tested By	HPE Performance Engineering	**Hardware/Firmware Availability**	Jul-2016
SPEC License #	3	**System Source**	Single Supplier
Test Location	Houston, TX, USA	**Test Date**	Mar 25, 2017
SERT 2 Efficiency Score	6.4		

Summary



Aggregate SUT Data	
Model	Hewlett Packard Enterprise ProLiant DL120 G7 (Historical System)
Number of Nodes	1
CPU Name	Intel(R) Xeon(R) CPU E3-1240 @ 3.30GHz
Total Number of Processors	1
Total Number of Cores	4
Total Number of Threads	8
Total Physical Memory	8.0 GB
Total Number of Memory DIMMs	2
Total Number of Storage Devices	1

Workload Efficiency Score			Idle Watts
CPU	**Storage**	**Memory**	
5.6	14.0	7.2	50.1

Figure 6: SERT 2.0.3 Report – html format

```
                                SERT (TM) Report
 Copyright (C) 2007-2017 Standard Performance Evaluation Corporation (SPEC).
                                All rights reserved.
The SERT license agreement prohibits the use of numerical information from this
report to make public comparisons promoting the use of one product over
another.

                                HEWLETT PACKARD ENTERPRISE PROLIANT DL120 G7 (HISTORICAL SYSTEM)
                  Test Sponsor | Hewlett Packard Enterprise                |          Software Availability | Apr-2016
                     Tested By | HPE Performance Engineering               |  Hardware/Firmware Availability | Jul-2016
                  SPEC License # | 3                                      |             System Source | Single Supplier
                     Test Location | Houston, TX, USA                      |                Test Date | Mar 25, 2017
            SERT 2 Efficiency Score | 6.4                                 |                          |

                                        Summary

                                    AGGREGATE SUT DATA
                                Model | Hewlett Packard Enterprise ProLiant DL120 G7
                                      | (Historical System)
                          Number of Nodes | 1
                                 CPU Name | Intel(R) Xeon(R) CPU E3-1240 @ 3.30GHz
               Total Number of Processors | 1
                  Total Number of Cores | 4
                 Total Number of Threads | 8
                   Total Physical Memory | 8.0 GB
            Total Number of Memory DIMMs | 2
          Total Number of Storage Devices | 1

    Workload Efficiency Score       | Idle Watts
   CPU     |  Storage   |   Memory   |
---------- | ---------- | ---------- | ----------
   5.6     |    14.0    |    7.2     |    50.1
```

Figure 7: SERT 2.0.3 Report – txt format

5.4. Validation / Verification

The SERT software components implement software checks wherever possible to increase information accuracy, verify user input, monitor run-time data collection, and validate results. The intent is to improve accuracy, remedying user errors and to prevent invalid data being reported.

When conditions or results do not meet specific criteria, warnings are displayed and error messages will appear in the SERT reports.

Examples of compliance checking are:

- Java class files in the SERT jar files have not been modified in any way.
- Host and Client JVMs must be 64-bit.
- There are no failures in the transactions that are run as part of the SERT execution.
- Minimum temperature of the SUT must be >= 20 °C.
- <2% error readings recorded by the temperature sensor for entire run, at least one sample per interval
- Non-editable sections of results.xml file must not have been edited.
- Configuration file (default name: “config-all.xml”) is compliant.
- Results must have been obtained using a released version of SERT to be compliant.
- The power analyzer used must be an accepted device per the PTDaemon. The test date must be within 1 year of calibration date. The analyzers used for the test must be the same as described in the test-environment.xml file.
- <=1% error readings for Power, <=2% for Volt, Ampere, and PF, measured only during recording period
- For each measurement interval on each analyzer, no more than 5% of all samples can have uncertainty >1%, no more than 1% can have unknown uncertainty, average uncertainty of all samples <= 1%.
- The temperature sensor must be an accepted device per the PTDaemon. The sensors used for the test must be the same as described in the test-environment.xml file.
- The results must meet the target throughput (within +2%/-4% for 90-100%, within +/- 2% for others).
- Throughput variation between hosts (less or equal 10%)

All the SERT software components will perform validation checks within the domain of their functions, e.g. warnings of connection problems, log measurement errors and out-of-range conditions, warning the user of missing or incomplete information and to check the validity of some entered data.

5.5. Logging

If there are any errors or warnings generated while SERT is running, they are saved in the log files. Depending on where the error/warning occurs, the log files are created on the Controller system or the SUT.

SERT package includes two scripts, one for Windows based systems and another for Linux based systems. The two script files are below:

`-collectlogfiles.bat` is intended to be executed on a Microsoft Windows Operating System.

`-collectlogfiles.sh` is intended to be executed on a Linux, Solaris, or AIX system.

These scripts collect all the log files on the respective system and create a ZIP or tar.gz file respectively in the SERTlog subdirectory under the main SERT installation directory. The file is name is formatted as:

`<hostname>_YYYY-MM-DD_HHMM.<zip/tar.gz>` (where YYYY is the year, MM is the month, DD is the day, and HHMM represents the hours and minutes count.)

The ZIP/tar.gz file can be provided to SPEC for analysis and root cause analysis of the issue.

6. Worklet Design Guidelines

In order to achieve consistent results from all worklets and a broad coverage of technologies the following guidelines have been observed:

- Each worklet is adjustable to different performance levels, in particular to some predefined levels between 100% (maximum load) and 0% (idle).
- Each worklet automatically calibrates load levels based on the maximum performance measured by the tool, independent of user action.
- Precompiled binaries of the test programs are used.
- The worklets scale with the available hardware resources. More resources should result in an appropriate increase in the performance score; e.g. more processor/memory/disk capacity or additional processor/memory/disk modules yield a better result in the performance component of the efficiency rating.
- Each worklet is portable code that follows all SPEC rules for licensing, reuse, and adaptation.
- The worklets are both architecture- and OS-agnostic or accommodate different architectures and/or OSes by using mildly different code paths.
- The work accomplished by each worklet is clearly identifiable as “important” but is not required to cover “all important” types of work.

In order to follow these guidelines, most worklets use the concept of batches of discrete work, where each batch constitutes a transaction. The different load levels will be achieved by scheduling the required number of transactions, after an initial calibration phase that estimates the 100% performance level.

SERT worklets are not designed to explicitly exercise General Purpose Graphics Processing Units (GPGPUs).

6.1. CPU Worklets

The defining characteristics of the CPU worklets are:

- The worklet requires consistent processor characteristics per simulated “user” regardless of number of processors, cores, enabled threads, etc.
- At the 100% load level, the performance bottleneck is the processor subsystem.
- The worklet’s performance should increase with more processor resources, including the number of processors, the number of cores, possibly the number of logical processors, increased frequency, larger available cache, lower latency, and faster interconnect between CPU sockets.

A more detailed discussion of these worklets follows in the next chapter; generally, the group includes code patterns used in compression, cryptography, numerical processing, sorting, and the processing of HTML.

6.2. Memory Worklets

The defining characteristics of the memory worklets are:

- The worklet contains consistent memory access characteristics per simulated “user” regardless of size and number of memory DIMMs.
- At the 100% load level, the performance bottleneck is the memory subsystem.
- The worklet's performance should measure a higher (better) performance score with improved memory characteristics (e.g. higher bandwidth, lower latency, total memory size), and
- The worklets as a group should reflect a combination of random and sequential reads and writes, and small and large memory accesses.

As discussed in more detail below, the worklets in this category consist of a memory throughput workload, and a memory capacity workload.

6.3. IO Worklets

Disk and Network IO components are a key part of any well-rounded picture of system performance and power. Worklets in this category are intended to reflect the performance and power usage of modern storage subsystems with higher performance, larger capacity, and extensive reliability and availability features.

SPEC recognizes that some of the items in the next two sections may not be reasonable or practical to test or measure in a meaningful way. In those cases, the use of "Configuration Power/Performance Modifier" to compensate for the extra power draw associated with extra functionality is recommended.

The measurements of power and performance of either optional add-in storage controller cards or server blade enclosure storage are not in the scope of the SERT.

6.3.1. Network IO Worklets

No Network IO worklet is included in the SERT SUT, and the main reasons for this decision are:

- The cost of testing all reasonable external test system configurations
- Initial measurements show that there are no significant differences in power utilization between 100% and 0% network utilization for today's technology.

6.3.2. Storage IO Worklets

The SERT does include Storage IO Worklets, whose key characteristics are:

- The worklets reflect consistent IO characteristics per simulated "user" regardless of system size and number of disks or the installed memory.
- The worklets consist of a combination of random and sequential accesses, reads and writes, and small and large IOs.
- At the 100% load level, the performance bottleneck is the storage subsystem.
- The worklets should score a higher (better) performance result for higher bandwidth and lower latency.
- The worklets are limited to testing individual internal storage devices only. RAID arrays and external storage devices are not supported.

The SERT bypasses operating system caches and requires disk and controller caches to be disabled, and therefore does not measure any performance or power benefits provided by these caches.

6.4. Idle Worklet

During idle measurements, the SUT must be in a state in which it is capable of completing workload transactions. Therefore, the idle worklet is treated in a manner consistent with all other worklets, with the exception that no transactions occur during the measurement interval.

7. Worklet Details

Each worklet supports a variety of configuration parameters for conducting experiments, nonetheless the SERT suites defines the values of these parameters for compliant runs. The following table lists the current worklets and their load levels for each of the workloads.

Workload	Load Level	Worklet Name
CPU	100%, 75%, 50%, 25%	Compress
		CryptoAES
		LU
		SHA256
		SOR
		SORT
	100%, 87.5%, 75%, 62.5%, 50%, 37.5%,25%, 12.5%	SSJ
Memory	Full, Half	Flood3
	Max, Base	Capacity3
Storage	100%, 50%	Random
		Sequential
Idle	idle	Idle

7.1. CPU Worklet: Compress

7.1.1. General Description

The Compress workload implements a transaction that compresses and decompresses data using a modified Lempel-Ziv-Welch method (LZW). Essentially, it finds common substrings and replaces them with a variable size code. This is both deterministic and done on the fly. Thus, the decompression procedure needs no input table, but tracks the way the table was built. The algorithm is based on "A Technique for High Performance Data Compression", Terry A. Welch, IEEE Computer Vol. 17, No. 6 (June 1984), pp 8-19.

7.1.2. Sequence Execution Methods

Graduated Measurement Sequence

7.1.3. Metric

Transactions Per Second

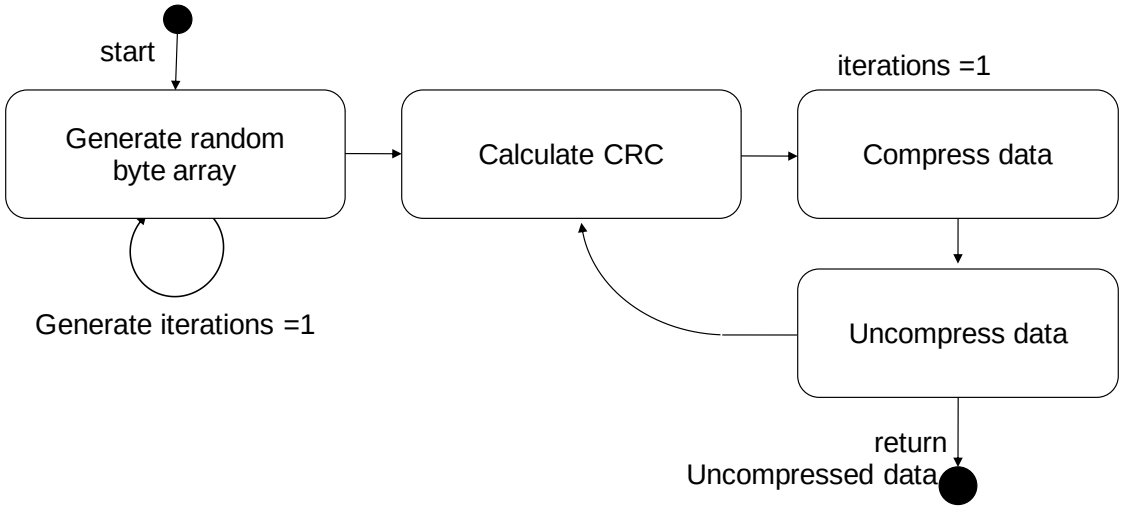
7.1.4. Required Initialization

A constant size byte array is generated on the fly before for each transaction execution. The contents of the byte array are randomly generated.

7.1.5. Configuration Parameters

`size`	Size of the input byte array for each transaction execution.
`enable-idc`	Enables/disables memory scaling using input data caching (IDC). Must be set to false.
`iterations`	Number of executions per transaction
`debug-level`	Value governs the volume of debug messages printed during execution.
`input-generate-iterations`	Number of random byte array assignment iterations

7.1.6. Transaction Code



7.2. CPU Worklet: CryptoAES

7.2.1. General Description

The CryptoAES workload implements a transaction that encrypts and decrypts data using the AES (or DES) block cipher algorithms. The SERT suite uses AES with CBC and no PKCS5 padding. Encryption and decryption are done using the Java Cryptographic Extension (JCE) framework, and the Cipher class, in particular.

7.2.2. Sequence Execution Methods

Graduated Measurement Sequence

7.2.3. Metric

Transactions Per Second

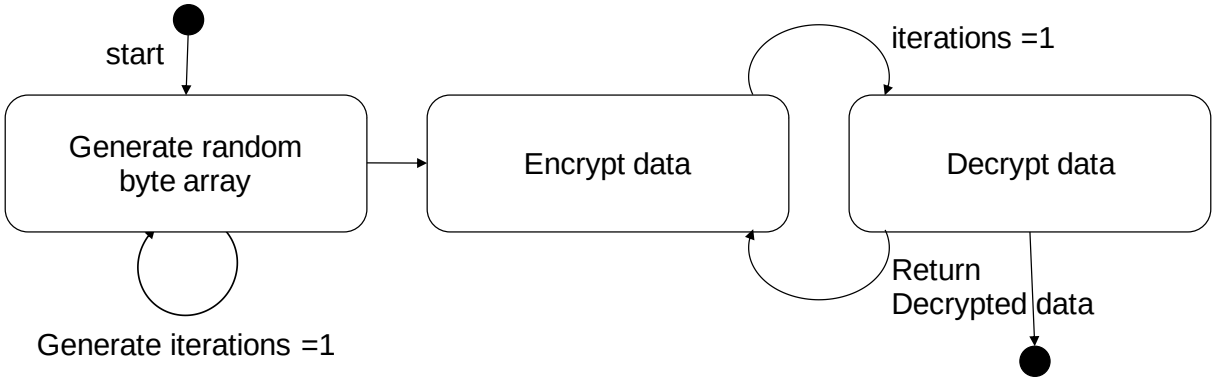
7.2.4. Required Initialization

A constant size byte array is generated on the fly before for each transaction execution. The contents of the byte array are randomly generated.

7.2.5. Configuration Parameters

size	Size of the input byte array for each transaction execution
key-generator	Key generator algorithm (AES or DESede)
key-size	Key size. (128 for AES, 168 for DES)
algorithm	Encryption algorithm. (E.g., AES/CBC/NoPadding, AES/CBC/PKCS5Padding, DESede/CBC/NoPadding, DES/CBC/PKCS5Padding)
Level	Number of times to perform the encryption
enable-idc	Enables/disables memory scaling using input data caching (IDC). Must be set to false.
iterations	Number of executions per transaction
debug-level	Value governs the volume of debug messages printed during execution.
input-generate-iterations	Number of random byte array assignment iterations

7.2.6. Transaction Code



7.3. CPU Worklet: LU

7.3.1. General Description

The LU workload implements a transaction that computes the LU factorization of a dense matrix using partial pivoting. It exercises linear algebra kernels (BLAS) and dense matrix operations. The algorithm is the right-looking version of LU with rank-1 updates. (Adapted from the NIST-developed SciMark benchmark).

7.3.2. Sequence Execution Methods

Graduated Measurement Sequence

7.3.3. Metric

Transactions Per Second

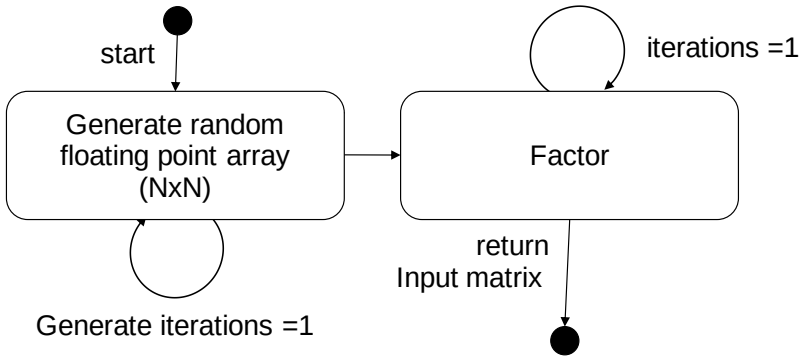
7.3.4. Required Initialization

A constant size matrix of floating point numbers is generated on the fly before for each transaction execution. The contents of the matrix are randomly generated.

7.3.5. Configuration Parameters

matrix-dimen	Dimension of the input floating point matrix for each transaction execution (NxN)
enable-idc	Enables/disables memory scaling using input data caching (IDC). Must be set to false.
iterations	Number of executions per transaction
debug-level	Value governs the volume of debug messages printed during execution.
input-generate-iterations	Number of random matrix assignment iterations

7.3.6. Transaction Code



7.4. CPU Worklet: SHA256

7.4.1. General Description

Hashing and encryption/decryption are two pillars of modern computer security. The SHA-256 workload utilizes standard Java functions to perform SHA-256 transformations on a byte array. This byte array is perturbed by one byte for each transaction.

7.4.2. Sequence Execution Methods

Graduated Measurement Sequence

7.4.3. Metric

Transactions Per Second

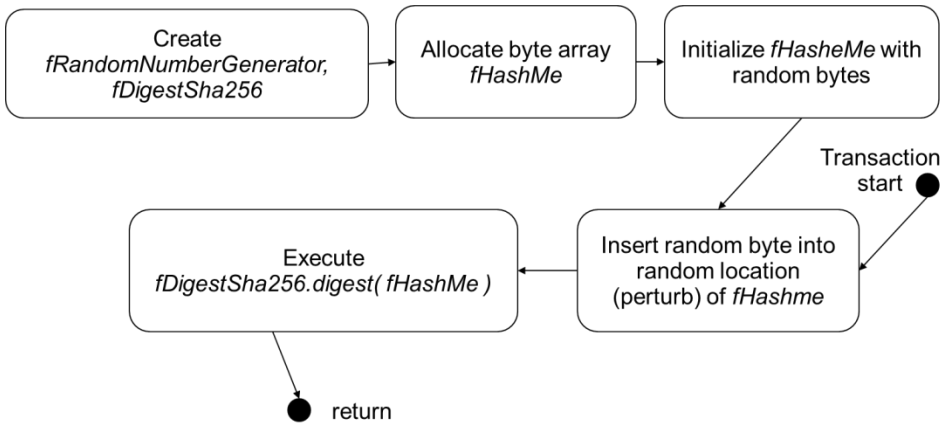
7.4.4. Required Initialization

None

7.4.5. Configuration Parameters

`Debug-level`	Detailed diagnostic information can be enabled through the debug parameter. Valid values are 0 = no additional debug information (default), -1 = debug information turned on.

7.4.6. Transaction Code



7.5. CPU Worklet: SOR

7.5.1. General Description

The Jacobi Successive Over-relaxation (SOR) workload implements a transaction that exercises typical access patterns in finite difference applications, for example, solving Laplace's equation in 2D with Drichlet boundary conditions. The algorithm exercises basic "grid averaging" memory patterns, where each `A(i,j)` is assigned an average weighting of its four nearest neighbors. Some hand-optimizing is done by aliasing the rows of `G[][]` to streamline the array accesses in the update expression. (Adapted from the NIST-developed SciMark benchmark).

7.5.2. Sequence Execution Methods

Graduated Measurement Sequence

7.5.3. Metric

Transactions Per Second

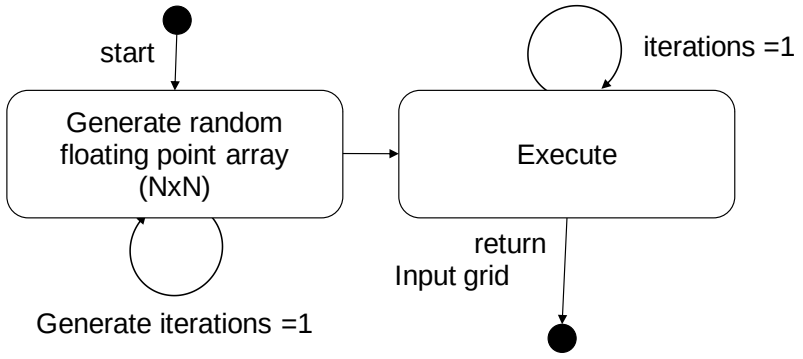
7.5.4. Required Initialization

A constant size grid of floating point numbers is generated on the fly before for each transaction execution. The contents of the grid are randomly generated.

7.5.5. Configuration Parameters

`grid-dimen`	Dimension of the input floating point grid for each transaction execution (NxN)
`enable-idc`	Enables/disables memory scaling using input data caching (IDC). Must be set to false.
`iterations`	Number of executions per transaction
`debug-level`	Value governs the volume of debug messages printed during execution.
`input-generate-iterations`	Number of random grid assignment iterations

7.5.6. Transaction Code



7.6. CPU Worklet: SORT

7.6.1. General Description

Sorting is one of the most common and important operations in computing. The SORT worklet sorts a randomized 64-bit integer array during each transaction. The worklet uses the java.util.Arrays.sort( ) API to perform the sort -- the implementation of this API may vary in different JVM versions.

7.6.2. Sequence Execution Methods

Graduated Measurement Sequence

7.6.3. Metric

Transactions Per Second

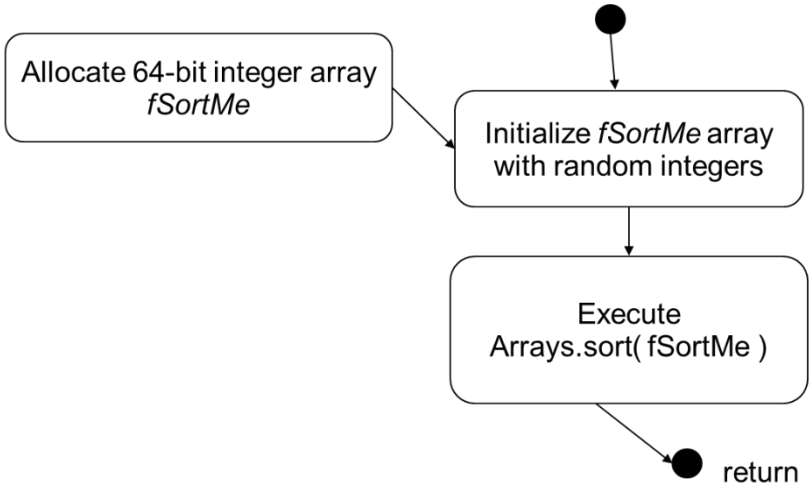
7.6.4. Required Initialization

None

7.6.5. Configuration Parameters

`Debug-level`	Detailed diagnostic information can be enabled through the debug parameter. Valid values are 0 = no additional debug information (default), -1 = debug information turned on.

7.6.6. Transaction Code



7.8. CPU Worklet: SSJ

7.8.1. General Description

SSJ is a simulated Online Transaction Processing (OLTP) workload, and represents a Server Side Java application. It is based on the SSJ workload in SPECpower_ssj2008, which was based on SPECjbb2005, which was inspired by the TPC-C specification; however, there are several differences between all of these workloads, and SSJ results are not comparable to any of these benchmarks.

The SSJ Worklet exercises the CPU (s), caches, and memory of the SUT. The peak throughput level is determined by maximum number of transaction of the above type the system can perform per second. Once the peak value of the transactions is determined on a given system, the worklet is run from peak (100%) down to the system idle in a graduated manner.

The performance of the Worklet depends on the combination of the processor type, number of processors, their operating speed, and the latency and bandwidth of the memory subsystem of the system.

SSJ includes 6 transactions, with the approximate frequency shown below:

- New Order (30.3%) – a new order is inserted into the system
- Payment (30.3%) – records a customer payment
- Order Status (3.0%) – requests the status of an existing order
- Delivery (3.0%) – processes orders for delivery
- Stock Level (3.0%) – finds recently ordered items with low stock levels
- Customer Report (30.3%) – creates a report of recent activity for a customer

7.8.2. Sequence Execution Methods

Graduated Measurement Sequence

7.8.3. Metric

Transactions Per Second

7.8.4. Required Initialization

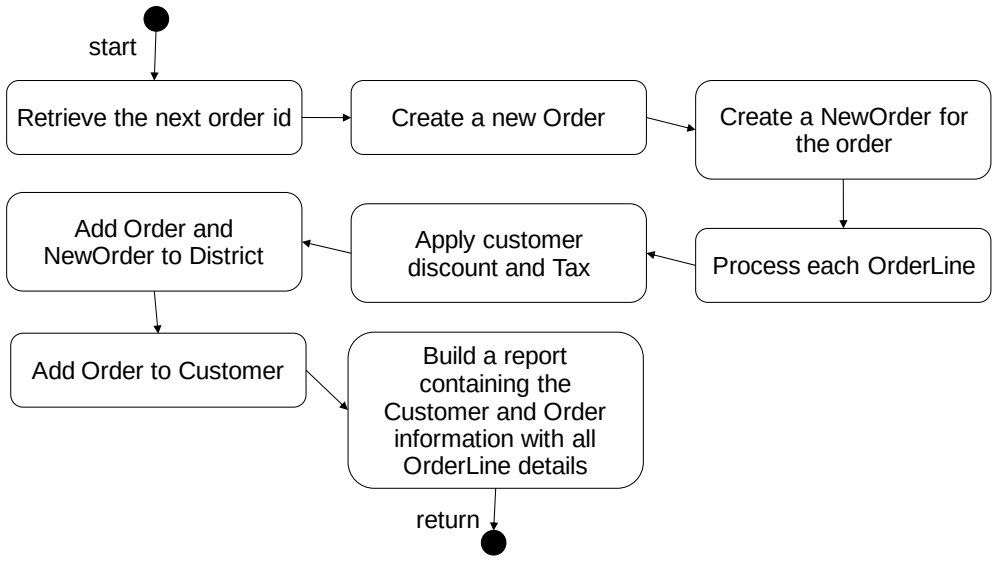
Each user represents a warehouse. During initialization, each warehouse is populated with a base set of data, including customers, initial orders, and order history.

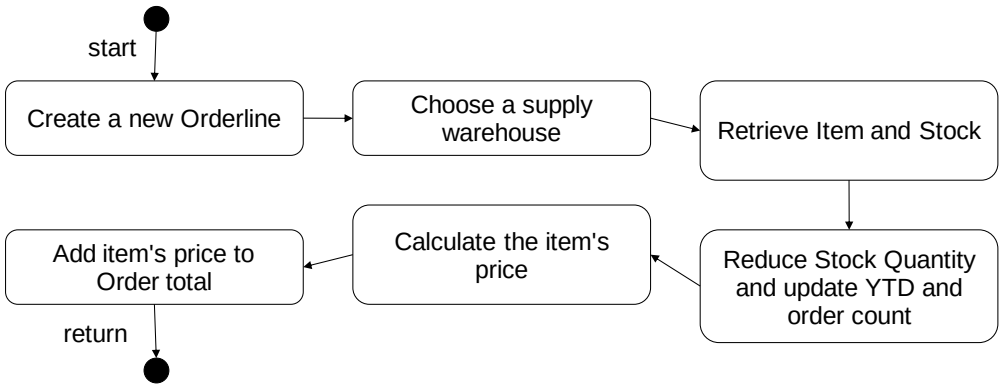
7.8.5. Configuration Parameters

The SSJ workload does not have any supported configuration parameters.

7.8.6. New Order Transaction

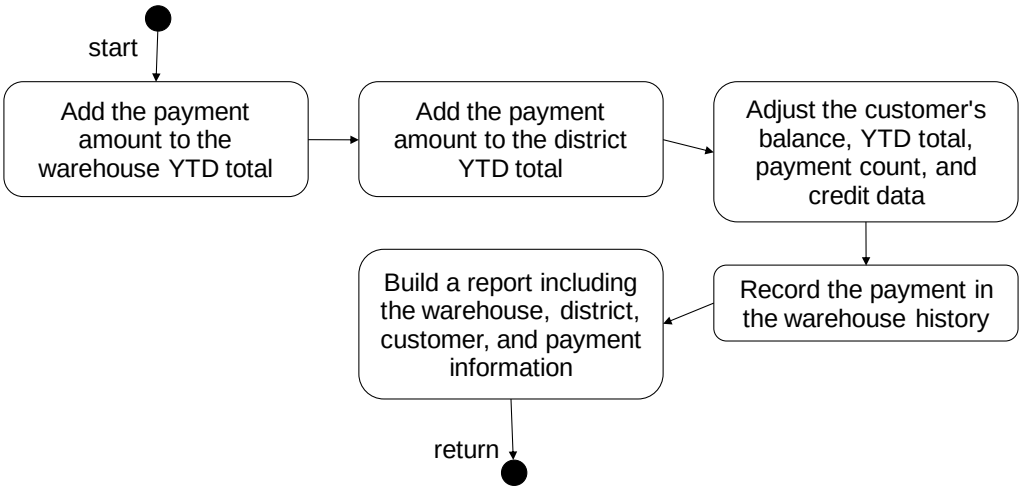
The input for a New Order Transaction consists of a random district and customer ID in the user's warehouse, and a random number of orderlines between 5 and 15.





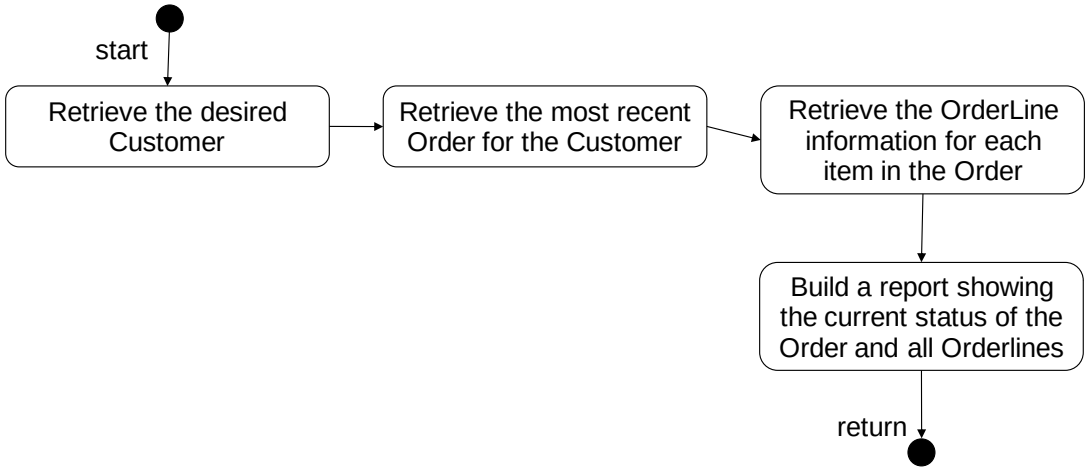
7.8.7. Payment Transaction

The input for a Payment Transaction consists of a random district from the user's warehouse, a random customer id or last name (from either the user's warehouse or a remote warehouse), and a random payment amount.



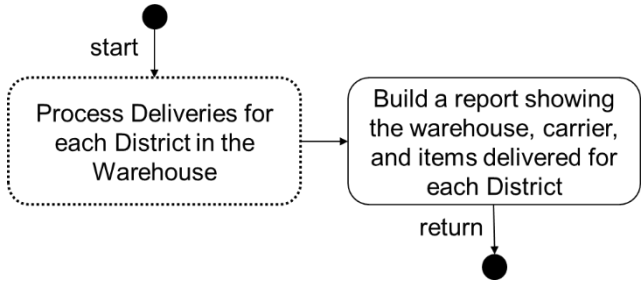
7.8.8. Order Status Transaction

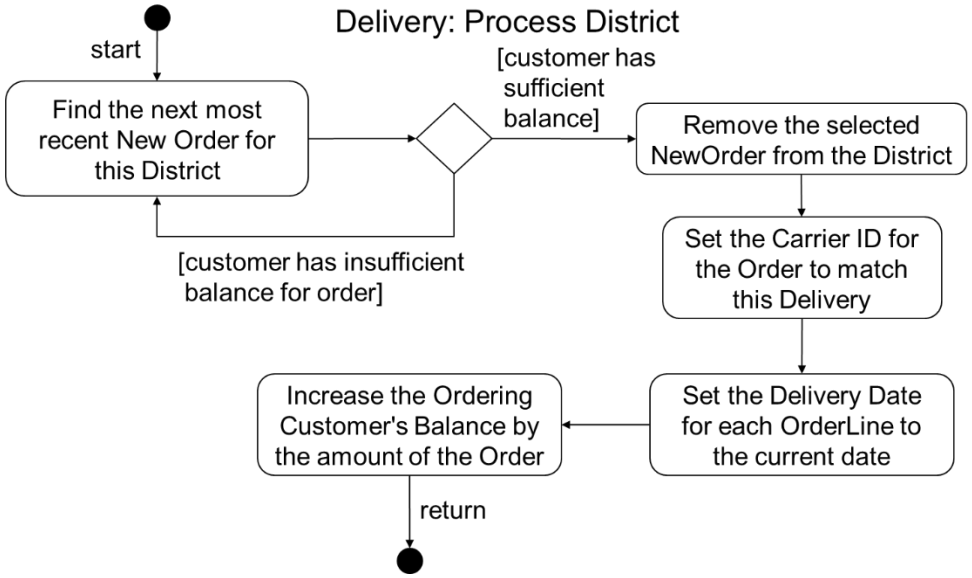
The input for an Order Status Transaction consists of a random district and either a customer ID or last name from the user's warehouse.



7.8.9. Delivery Transaction

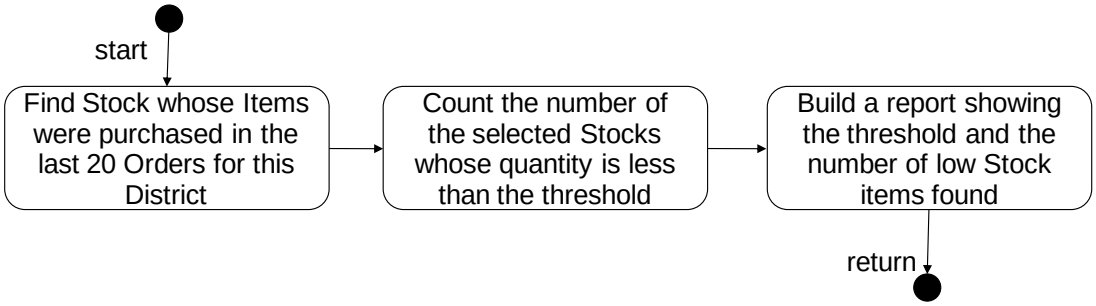
The input for a Delivery Transaction is a random carrier ID.





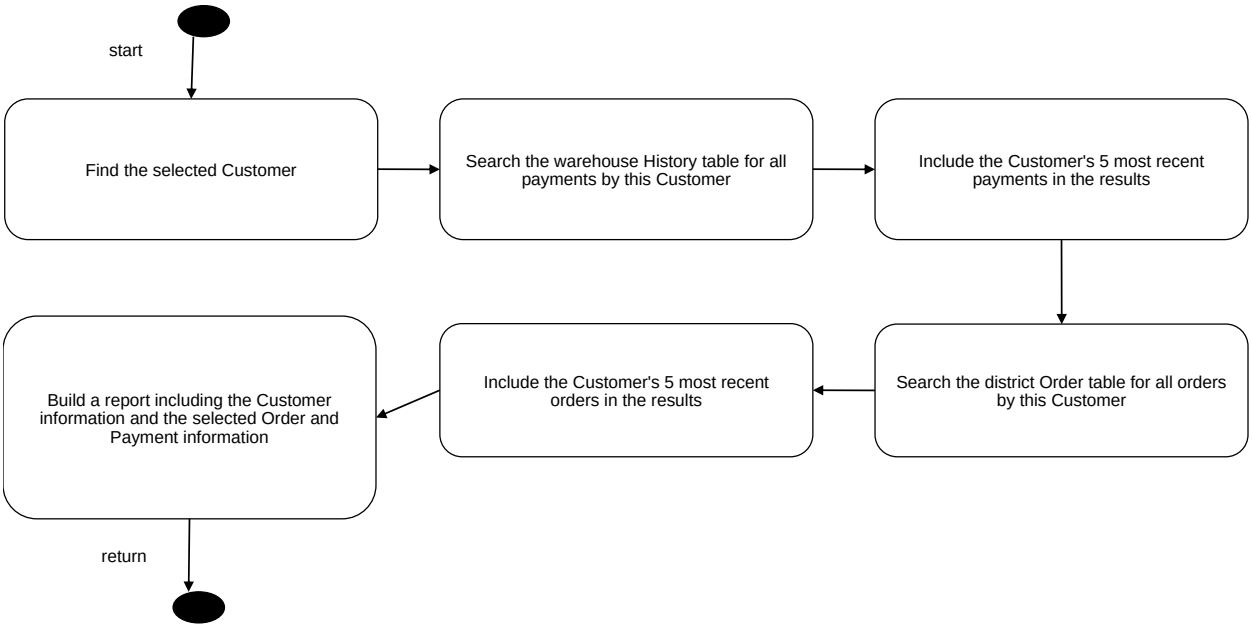
7.8.10. Stock Level Transaction

The input for a Stock Level transaction is a random district from the user's warehouse and a random “low level” threshold between 10 and 20.



7.8.11. Customer Report Transaction

The input for a Customer Report transaction consists of a random district from the user's warehouse and a random customer ID or last name (from either the user's warehouse or a remote warehouse).



7.9. Memory Worklet: Flood3

7.9.1. General Description

The Flood3 workload is based upon STREAM, a popular benchmark that measures memory bandwidth across four common and important array operations. For the *long* (64-bit) integer arrays used in Flood3, the following amounts of memory are involved per assignment:

1. **COPY:** *a(i) = b(i)*
   -- 8 bytes read + 8 bytes write per assignment = 16 bytes / assignment
2. **SCALE:** *a(i) = k * b(i)*
   -- 8 bytes read + 8 bytes write per assignment = 16 bytes / assignment
3. **ADD:** *a(i) = b(i) + c(i)*
   -- 16 bytes read + 8 bytes write per assignment = 24 bytes / assignment
4. **TRIAD:** *a(i) = b(i) + k * c(i)*
   -- 16 bytes read + 8 bytes write per assignment = 24 bytes / assignment

The Flood3 score is based upon the aggregate system memory bandwidth calculated from the average of these four tests multiplied by the amount of physical memory installed in the SUT. While Flood3 is based upon STREAM, it uses no STREAM code and is implemented wholly in Java.

Flood3 enhances STREAM in a variety of important ways:

1. Flood3 is designed to fully exploit the memory bandwidth capabilities of modern multi-core servers. Flood3 is multi-threaded and threads are scheduled to operate concurrently during bandwidth measurements ensuring maximum throughput and minimizing result variability.
2. Flood3 requires little to no user configuration, yet automatically expands the data set under test to fully utilize available memory.

Measuring aggregate system memory bandwidth on large servers with many cores and multiple memory controllers is challenging. In particular, run-to-run variability is often unmanageable with existing memory bandwidth benchmarks. Flood3 minimizes run-to-run variation by taking three memory bandwidth tests back-to-back and discarding the first and last tests. This ensures that all threads are running under fully concurrent conditions during the middle measurement which is used in Flood3 scoring calculations.

Flood3 scores scale with a SUT's aggregate memory bandwidth as well as with the SUT's physical memory configuration. CPU, storage and network performance have little to no impact on Flood3 scores.

The Flood3 worklet automatically adjusts the amount of memory under test to fully utilize installed DRAM, and also adjusts the number of iterations to improve run-to-run consistency. As a result, the run time will vary depending on the system configuration details, including the amount of physical memory and the overall processor thread count.

7.9.2. Sequence Execution Methods

MinIterationsDirectorSequence – Flood3 is executed for a given set of iterations specified within *config-all.xml.*

7.9.3. Metric

By default, *Flood3* runs the four memory bandwidth tests that are also part of STREAM, but were independently implemented (i.e. no code was utilized from STREAM). Each test returns a score that is based upon the measured raw memory bandwidth:

```
score = bandwidthGbs
```

These scores are scaled (based upon load level), aggregated, and averaged. The load level is 100% in *Flood_Full* and 50% in *Flood_Half* (which uses data sets reduced down to 50% of *Flood_Full*):

```
//measurement:

result = Math.abs( RunCopySystem() * fLoadLevel );

result += Math.abs( RunScale() * fLoadLevel );

result += Math.abs( RunAdd() * fLoadLevel );

result += Math.abs( RunTriad() * fLoadLevel );

result /= 4; // take average
```

In turn, the SERT suite aggregates these results for all *Flood3* threads. Typically, *Flood3* achieves raw memory bandwidth results around 60%-90% that of highly tuned, multithreaded (OMP) C++ STREAM binaries, but usually with much better reproducibility. Furthermore, *Flood3* utilizes nearly all available memory whereas STREAM uses much smaller data sets.

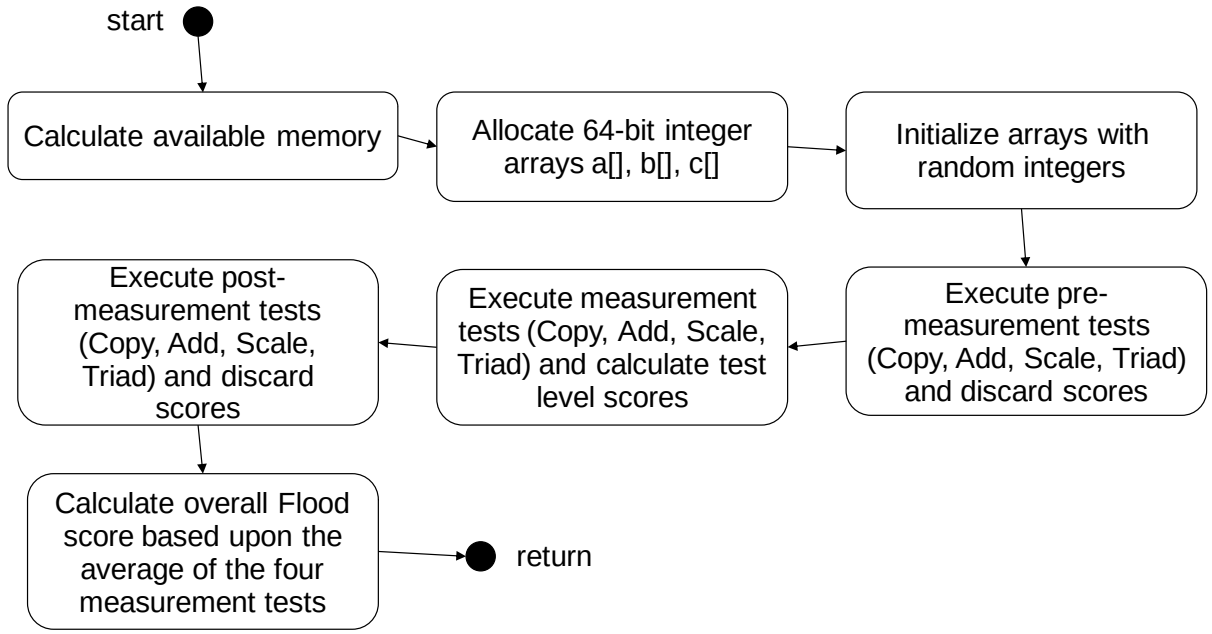
7.9.4. Required Initialization

Flood3 calculates the amount of memory available to the thread and creates three 64-bit (*long*) integer arrays, a[], b[], and c[], to completely utilize all available space. These arrays are initialized with random data. To ensure full load concurrency during bandwidth measurements, a complete set of pre-measurement tests is launched prior to the measurement period followed by the post-measurement tests. Only the test results for the measurement period are utilized for Flood3 score generation.

7.9.5. Configuration Parameters

memory-under-test	The default value of "-1 MB" turns on automatic configuration of the data set size. However, the user can override this behavior and explicitly define the amount of memory to test per JVM. Valid values are (san quotation marks): "200 MB", "1.1 GB", "10000000 B".
iterations	Flood3 internally iterates the number of memory bandwidth tests based upon the value of the iterations parameter. The default is -1 (determine automatically).
debug-level	Detailed diagnostic information can be enabled through the debug parameter. Valid values are 0 = no additional debug information (default), 1 = debug information turned on, 2 = detailed debug information.
return-bandwidth	The raw, aggregate system memory bandwidth calculated by Flood3 can be obtained by setting the parameter return-bandwidth to "true" in which case Flood3 will return measured memory bandwidth instead of a score. The default value is "false".

7.9.6. Transaction Code



7.10. Memory Worklet: Capacity3

7.10.1. General Description

The Capacity3 worklet implements a transaction that exercises Java's XML validation package javax.xml.validation. Using both SAX and DOM APIs, an XML file (.xml) is validated against an XML schemata file (.xsd). To randomize input data, an algorithm is applied that swaps the position of commented regions within the XML input data.

Memory scaling in Capacity3 is done through a scheme known as input data caching (IDC). In IDC, the universe of possible input data (here, randomized XML file data) is pre-computed and then cached within memory before the start of the workload. During workload execution, the input data for a particular transaction instance is then chosen randomly and retrieved from this cache rather than computed on the fly. In order for the data store to completely fit into the data cache physical memory of about twice the data store size is needed.

The Capacity3 worklet executes two intervals:

- **Capacity3_Base** has a fixed data store size of 4GB
- **Capacity3_Max** has a dynamic data store size calculated as half the size of available physical memory, i.e. the maximum data store size that typically fits into data cache without causing cache misses. This maximum data store size is not limited to 1TB as it was in Capacity2.

Cache size is computed as:

`MaxHeapSize * data-cache-to-heap-ratio` (set to 0.6 in SERT)

While this worklet does contain transactions that are memory oriented, there is still a component that is influenced by CPU performance.

7.10.2. Sequence Execution Methods

A Modified Parameters Sequence. Each interval consists of a No Delay Series where the parameter data-store-size changes with each interval.

7.10.3. Metric

Metric formula for Capacity3

```
Transactions Per Second * sqrt(data-store-size)
```

The overall score will scale with memory size.

Note: In SERT 2.0 the Capacity worklet was replaced with the Capacity3 worklet which used an updated method to run the worklet and calculate the metric.

7.10.4. Required Initialization

At initialization time, both XML and XML schemata files are read in from disk and saved in a buffer for future use. (There will be no further disk IO once this is completed.) IDC initialization follows during which all possible input data sets are pre-computed and cached in memory. For each input data set, a randomization algorithm is applied to the original XML data to create variations in parsing without modifying file size or complexity.

7.10.5. Configuration Parameters

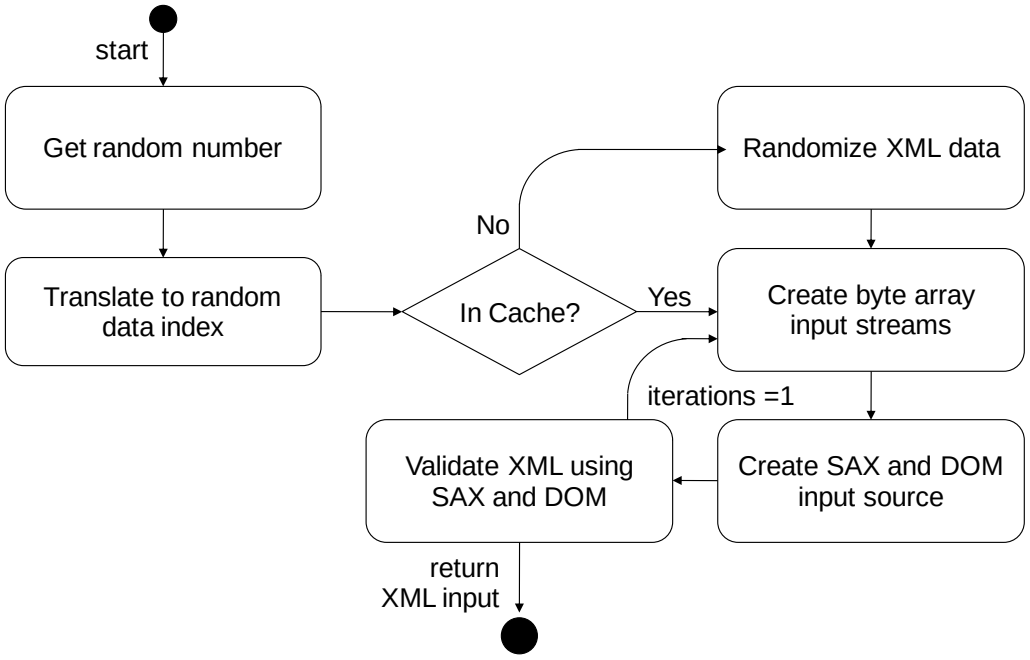
xml-schema-dir	Specifies the directory of the XML schema file
xml-schema-file	Specifies the name of the XML schema file
xml-dir	Specifies the directory of the XML file
xml-file	Specifies the name of the XML file

enable-idc	Enables/disables memory scaling using input data caching (IDC). Must be set to false.
iterations	Number of executions per transaction.
debug-level	Value governs the volume of debug messages printed during execution.
input-generate-iterations	Number of XML file randomization iterations.

Additional IDC configuration parameters:

store-type	Specifies the algorithm to use in generating data when a cache miss occurs
locality-distribution	Specifies the probability distribution to use when randomly choosing input data indices
data-store-size	Specifies the size of the universe of possible input data
data-cache-size	Specifies the size of the input data cache
data-cache-report-interval	Governs the frequency of output messages on cache hit/miss ratio
custom-score-policy	Specifies the algorithm to use in computing custom score reflecting cache size configuration.
data-cache-size-scale factor	Specifies the scaling factor to use in the DataCacheSizeMultiplierGB custom scoring algorithm
data-cache-to-heap-ratio	Ratio of cache size to JVM heap size used in automatic cache sizing

7.10.6. Transaction Code



7.11. Storage IO Workload

7.11.1. General Description

The Storage-Workload has four different transactions, two random and two sequential transaction-pairs. Each pair has a write and a read transaction.

7.11.2. Sequence Execution Methods

Graduated Measurement Sequence

7.11.3. Metric

Transactions (IO operations) per second.

7.11.4. Required Initialization

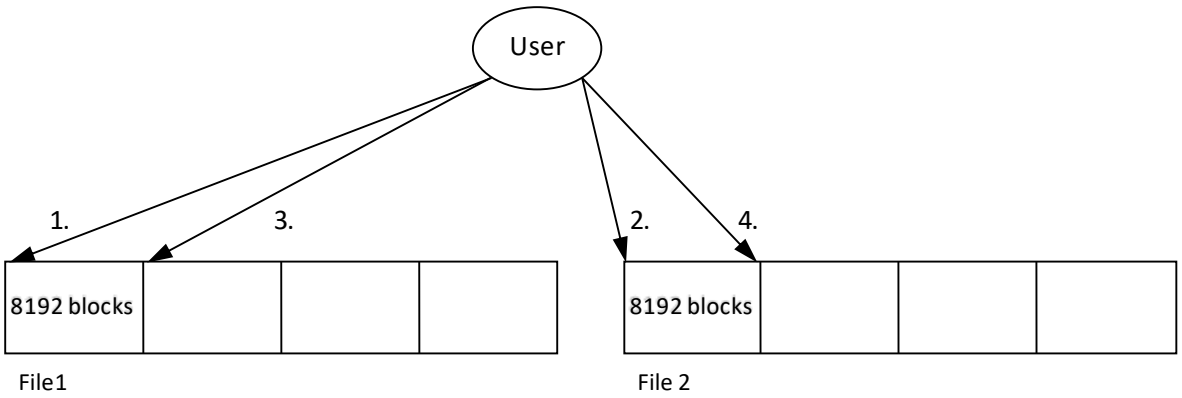
A set of files is created before execution of the transaction

7.11.5. Configuration Parameters

file-size-bytes	size of a file
file-per-user	number of files opened by each user
file-path	location of the files - In this example the path is “D:\data\”, please note that the files always reside in a subfolder called “data”.
sequential-max-count	amount of blocks that are accessed by the sequential transaction in one file before the next file is addressed

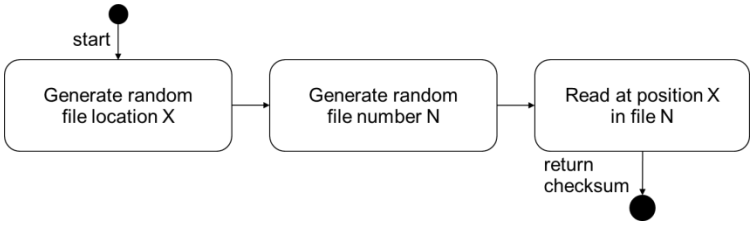
Example:

```
<file-size-bytes>1000000<file-size-bytes>

<file-path>D:\</file-path>

<file-per-user>2</file-per-user>

<sequential-max-count>8192</sequential-max-count>
```

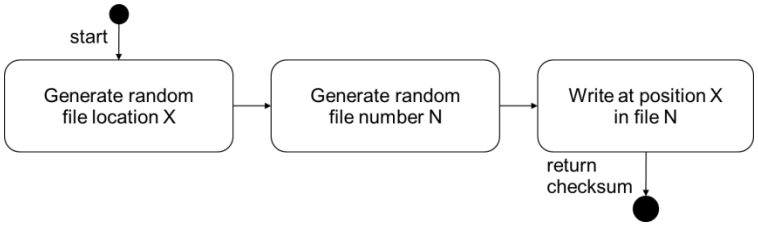


[File Example (2 files per user and max-count of 8192)]

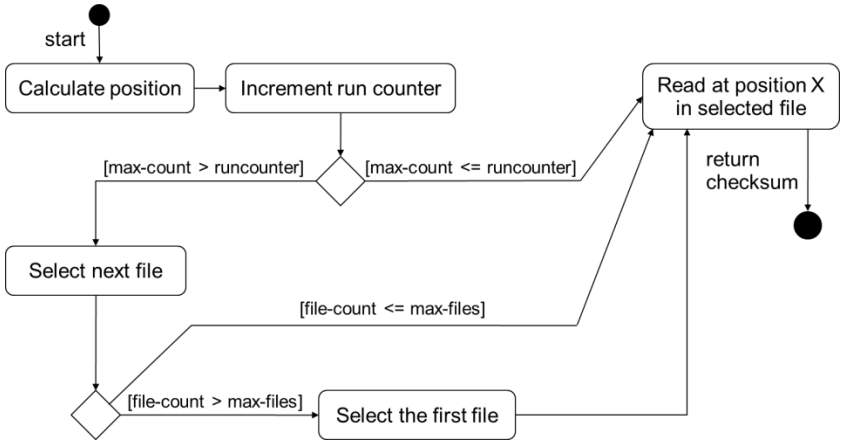
7.11.6. Transaction – Code 1 - RandomRead



7.11.7. Transaction – Code 1 - RandomWrite



7.11.8. Transaction – Code 2 – SequentialRead



7.11.9. Transaction – Code 2 – SequentialWrite