Intel® SoC Watch User Guide for Google Android* and Linux* OS

4th November 2019

Version 2020.1

Intel Corporation

www.intel.com

Legal Information

Contents

Legal Information

Intel, the Intel logo, Intel Atom, Intel Core, Intel Xeon Phi, VTune and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

Revision History

Revision Number	Description	Revision Date
2.0	Initial release for 2.0 product.	August 2015
2.1	Updated Building the Kernel Modules section	November 2015
2.1.1	Updated for 2.1.1 product release	December 2015
2.2	Updated for 2.2 product release and to combine Android*, and Linux* User's Guides	June 2016
2.2.1	Updated for 2.2.1 product release Update from `socperf1_2.ko` to `socperf2_0.ko` Bug fixes	June 2016
2.3.0	Updated for 2.3.0 product release	October 2016
2.3.1	Updated for 2.3.1 product release	April 2017
2.4	Update aligning available metrics and common reporting across all OSes, including Windows OS for the first time. Includes many new metrics for Intel® Core™ platforms.	November 2017
2.5	Added support for Intel platform code named Gemini Lake and other fixes.	February 2018
2.6	Enhancements include new hot key Alt-S, new --log option, modified metric groupings, and improved support in gfx metrics.	April 2018
2.6.1	Enhancements include new --log option and improved support in gfx metrics.	May 2018
2.7	Added average frequency report, new options (program-delay), new group names, support for Intel platforms code named Whiskey Lake and Amber Lake, fixed issues in ddr-bw, automation summary, and multiple pkg handling.	August 2018
2.8	Added support for new metrics (fpga-pwr, fpga-temp). Fixed errors in hw-cpu-cstate PC10 reporting (requires kernel version 4.4 or newer).	November 2018
2.9	Added trace file report grouping and informational messages.	January 2019
2.10	Added histogram for CPU frequency, added bandwidth metrics for FPGA, and bug fixes.	March 2019
2.11	Added option (-z) to automatically put the system into suspend state during a collection. Added reporting of power limits (PL1 and PL2) to feature -f pkg-pwr on Intel platform code named Apollo Lake. Includes support for Intel platform code named Cascade Lake-Xeon.	June 2019

Revision Number	Description	Revision Date
	Improves handling of unrecognized CPUs, reporting S-state when hibernation occurs, and other bug fixes.	
2019.12	Added support for Intel platform code named Ice Lake. Modified hw-cpu-pstate reporting.	September 2019
2019.13	Fixed issue in hw-cpu-pstate for Intel platform code named Ice Lake.	October 2019
2020.1	Added support for Intel platform code named Comet Lake.	November 2019

About Intel® SoC Watch

1

Intel® SoC Watch is a command line tool for monitoring and debugging system behaviors related to power consumption on Intel® architecture-based platforms. It reports active and low power states for the system/CPU/GPU/devices, processor frequencies and throttling reasons, wakeups, and other metrics that provide insight into the system's energy efficiency. The tool includes utility functions that include delaying the start of collection and launching an application prior to starting collection.

Data is collected from both hardware and OS sources. When using the default mode of collection, the tool collects data at normally occurring OS context-switch points so that the tool itself is not perturbing the system sleep states. Tool overhead when collecting during idle scenarios can be < 1%, however active workloads with a high-rate of context switching will increase the overhead. A minimum collection interval is used to control the rate of collection.

Intel SoC Watch writes a summary report file (.csv) at the end of collection on the system under analysis (target system), allowing immediate access to results. Additional result files can be specified including: an import file (.pwr) for Intel® VTune™Amplifier that can be used for visualization of correlated timelines for all the collected metrics with powerful zoom and filtering functions, and a time trace file (.csv) that can be viewed as a timelines in tools like Microsoft* Excel*.

Intended Audience

Use this document if you use Intel SoC Watch to analyze power consumption on a Linux or Android system.

Related Information

See the Intel® SoC Watch Release Notes for information on new features as well as known issues.

For online help, including information about importing results into Intel® VTune™Amplifier, see the Energy Analysis User Guide (https://software.intel.com/en-us/energy-analysis-user-guide).

Optimization Notice
Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

Installation

2

See the *Intel® SoC Watch Release Notes* for supported platforms and installation instructions.

Getting Started with Intel® SoC Watch

3

The following steps assume the Intel SoC Watch drivers and executables are installed. See the *Intel SoC Watch Release Notes* for instructions on how to install Intel SoC Watch.

Use the following steps to quickly collect processor C-state and P-state data for 60 seconds and import it into Intel VTune Amplifier for analysis.

Collect on Linux* OS

Set Up Collection on Linux

To collect directly on a Linux target, login as ‘root’. To collect remotely on a target device running Linux, use ssh to login to your target device as root.

Collection on Linux or Chrome OS

1. Load the device drivers. If the system is based on an Intel Atom® SoC, load the socperf3.ko driver. Then, load the socwatch2_10.ko driver on all platforms. Note that the socperf3.ko driver must be loaded before the socwatch2_10.ko driver.

   Previous versions of the socwatch2_x.ko driver (e.g. socwatch2_0.ko) should work but new collector support and/or bug fixes may be missing in the older drivers. If an older socwatch2_x.ko driver is used, some metrics may not be collected.

   On systems running Linux, issue the following commands:

```
insmod <path>/socwatch/socwatch_linux_[version]/drivers/socperf3.ko
insmod <path>/socwatch/socwatch_linux_[version]/drivers/socwatch2_10.ko
```

2. Navigate to the Intel SoC Watch directory.

```
cd <path>/socwatch/socwatch_linux_[version]
```

3. Setup the collection environment:

```
source ./setup_socwatch_env.sh
```

4. Create a results directory:

```
mkdir results
```

5. Collect data.

   For example, this command generates the test.csv, test.sw2 and test.pwr files in the results directory.

```
./socwatch -r vtune -m -f cpu-cstate -f cpu-pstate -t 60 -o ./results/test
```

6. View the summary results.

```
cat ./results/test.csv
```

7. To view results in Intel VTune Amplifier on your host system, copy the test.pwr file from the target to the host using scp. The following step assumes a Windows host.

```
scp root@<your_target_IP>:<path>/socwatch/<path>/results/test.pwr c:\results\.
```

Collect on Android OS

1. On the host system, establish a root adb shell on the target:

```
adb root
adb shell
```

2. If the target system has an Intel Atom® processor, load the socperf driver. The socperf driver must be loaded before the Intel SoC Watch driver:

```
insmod <path_to_socperf_driver>/socperf3.ko
```

NOTE
Make sure to use the latest version of the socperf driver.

3. Load the Intel SoC Watch driver:

```
insmod <path_to_socwatch_driver>/socwatch2_10.ko
```

4. Confirm the drivers are loaded:

```
lsmod
```

   Confirm the loaded drivers are included in the list of installed modules.

5. Setup the collection environment. This step assumes the default install directory was used.

```
cd /data/socwatch
source ./setup_socwatch_env.sh
```

6. Collect data and generate the test.csv, test.sw2, and test.pwr files in the results directory. This step assumes the /data/socwatch/results directory exists.

```
./socwatch -r vtune -m -f cpu-cstate -f cpu-pstate -t 60 -o ./results/test
```

7. Exit the adb shell:

```
exit
```

8. Use adb to pull the result files to the host:

```
adb pull /data/socwatch/results/test.csv c:\results
adb pull /data/socwatch/results/test.sw2 c:\results
adb pull /data/socwatch/results/test.pwr c:\results
```

Options Quick Reference

4

Invoke Intel SoC Watch with root privilege, using the following syntax:

```
socwatch <general options><post-processing options><collection options>
```

- Order of options does not matter unless specifically noted.
- Help is displayed if no option is specified.
- All features are not available on all systems, so the help text is dynamic, meaning it displays only the collection options that are supported by the system on which it is run. The metrics available differ because of changes in the system's hardware architecture support. This User's Guide contains a list of all metrics across all systems.
- You can specify feature names that are not available or not enabled on a particular system. When the tool starts, it will display console messages regarding features that cannot be collected, but collection will proceed if at least one feature is valid on that system.

Intel SoC Watch terminates data collection for one of three reasons (whichever occurs first):

1. the `--time` option was specified and the timer elapsed,
2. the `--program` option was used and the specified program exited,
3. a Ctrl-C interrupt was entered in the command window.

The location and name of the results files is displayed at the end of a collection. The summary report will be there with that name and a `.csv` extension. Raw data files and additional files based on post-processing options specified on the command line are located there as well, all with the same base name (default name is `SoCWatchOutput`).

> **NOTE**
> Result files are replaced if the same name is used for multiple collections.

General Options

The following options display information about the tool or system on which it is run.

Abbreviation	Option Name	Description
	`--export-help`	Write help output to JSON formatted file.
`-h`	`--help`	Display tool usage information and exit. The help shown is specific to the system on which it is run. Only metrics supported by the system architecture will be listed.
`-l`	`--log <filename>`	Redirect all console output, including errors, to specified file.
	`--print-fms`	Display CPU ID as Family.Model.Stepping and exit.
`-v`	`--version`	Display tool version information and exit.

Post-processing Options

The following options affect how results are reported and where they are stored.

Abbreviation	Option Name	Description
`-i`	`--input` *<filename>*	Specify the path and base filename (without extension) of an existing collection to generate additional reports. Use with the `-r` option to specify which types of reports.
`-o`	`--output` *<filename>*	Specify the base name for the output files from this collection. If this option is not specified, the files are written to the current working directory with base name `SoCWatchOutput`. Specifying `console` as the filename will cause the summary results to also write to stdout. If a name already exists, the previous results will be replaced.
`-r`	`--result` *<result_type>*	Specify the type of result to generate. This option can be repeated to get multiple types of reports. Following are the result types that can be specified: • `sum` Write summary reports to .csv file. [default] • `int` Write over-time data to _trace.csv file. • `vtune` Generate .pwr file for import to Intel VTune Amplifier. • `auto` Write summary results as a single line to file Automation_Summary.csv in current directory. Appends results, does not overwrite. If column headers for the new result changed, new headers will be inserted. Use to generate sets of data in a single file for comparison.

Collection Options

These options affect what is collected and how it is collected.

Abbreviation	Option Name	Description
`-f`	`--feature` *<name>*	Specify which metric to collect, choose from the group names or individual names listed in the tables below. This option can be repeated to collect multiple metrics in a single run. Most features can be collected simultaneously, exceptions noted in the table of feature names.
`-m`	`--max-detail`	Collect all data available for each feature specified. This will cause snapshot metrics to be sampled. Use of this option can increase tool overhead, so best used only when timeline of the data is needed or when collecting across system entry to hibernation. Without this option, the tool collects data at the minimum required by the data source for best accuracy. Data may be traced, sampled, or snapshot.

Abbreviation	Option Name	Description
		• Traced data is obtained at state transition points resulting in accurate summary and timeline results. • Sampled data is read at OS context switch points (or at timed intervals if polling option is used). This is less accurate as changes that take place between samples will not be measured. Metrics that come from hardware status/ state data must be sampled. • Snapshot data can be read at the beginning and end of the collection and the difference gives an accurate result with lowest overhead, but no timeline. Only metrics that come from hardware accumulators can be snapshot. The algorithm used to determine the collection method for each data type is as follows: If `-m` is specified: if the data can be traced, trace it; else sample it. If `-m` is not specified: if the data can be snapshot, snapshot it; else if the data can be traced, trace it; else sample it.
`-n`	`--interval` *<milliseconds>*	Specify the time in milliseconds that should pass before reading next hardware data sample (default 100 ms). For default collection mode, this is the minimum time between sampling at context switch points. When `--polling` option is used, this is actual time between samples. The minimum polling interval is 1ms. However, using low polling intervals will result in higher overhead and may fail to measure some metrics (e.g. bandwidths) with intervals shorter than the default.
	`--polling`	Make data collection occur at regular intervals rather than at context switch points. Use the `--interval` option to set the interval period (default: 100ms). Use of this option significantly increases perturbation of sleep states because it employs a timer which will interrupt sleep states, increase wakeup counts, and change timer resolution.
`-p`	`--program` *<application>* *<parameters>*	Specify the name of an executable to be started automatically prior to collection. The name can be followed by zero or more arguments that will be passed to the program.

Abbreviation	Option Name	Description
		NOTE This option must occur at the end of the command line, everything following the executable name will be given to it as arguments.
	`--program-delay <seconds>`	Specify number of seconds to wait before starting the program specified by `-p`. Has no effect if `-p` not used.
`-s`	`--startdelay <seconds>`	Specify number of seconds to wait before starting collection of data. If used with `-p` and `--program-delay`, this delay is applied after the program starts.
`-t`	`--time <seconds>`	Specify collection duration in seconds. Collection will stop when this time has elapsed unless Ctrl-C is entered or an executable specified with `--program` option exits prior to the specified duration.
`-z`		Automatically enter Suspend for the duration of the collection. Will automatically exit Suspend when the -t specified time expires. If system is woken from Suspend prior to the end of the duration, the collection will stop as well. If `--start-delay` is specified, it occurs prior to entering Suspend.

Feature Names (Individual)

The available feature names for the `--feature` option and their collection methods are listed below. You can specify multiple feature names individually or using group names described in the Feature Group Names section.

Note that every feature listed is not available on every platform supported by Intel SoC Watch. The `--help` option is dynamic, only showing features available for the platform on which it is run. Use it to determine which features are supported. You can specify unsupported features on the command line and the tool will simply display a message for those that cannot be collected, but continue with collection if there is at least one that is supported.

Collection methods are indicative of a metric's level of accuracy and overhead. Traced collection provides high accuracy along with precise transition points between states. Sampled collection is least accurate since transitions can occur which are never noted. Sampled data needs to be read at intervals throughout the collection period which increases tool overhead. Increasing the sampling rate (reading at closer intervals) will improve accuracy but increase overhead. Snapshot collection means the data comes from an accumulator so it can be collected only at the start and end of the collection period and give perfect accuracy. This gives accuracy and the lowest overhead. If the `--max-detail` (`-m`) option is given, the Snapshot metrics will instead be read at the same intervals as the Sampled metrics throughout the collection, so that you can generate a trace file to see how it changed overtime.

Name	Collection Methods	Description
`all-approx-bw`	Sampled	Estimated memory bandwidth for multiple SoC agents (concurrently), from hardware signal accumulators. Precise bandwidth can be collected for one agent at a time using other bandwidth features. Only one bandwidth or DRAM self-refresh metric can be specified at a time. Hardware counter overflow can occur if -m is not used for signal-based metrics. SoC transactions are counted and multiplied by 64 to form the estimates, which will be equal to or higher than the actual bandwidths. The estimate is high if partial or 32-byte transfers actually occurred. On 22nm Intel Atom® Processor-based SoC for Tablets and 2-in-1s (formerly code named Bay Trail), this feature provides the following estimated bandwidths: • Module 0 and Module 1: the bandwidth from CPU Module 0 and Module 1 to the DDR • GFX: the bandwidth from the graphics component to the DDR • Display: the bandwidth from the display controller to the DDR • ISP: the bandwidth from the camera image processor to the DDR • VED: the bandwidth from the video encode and decode components to the DDR • IO: the bandwidth between the north and south clusters On systems code named Cherry Trail, and Intel Atom Processor Z35XX (formerly code named Moorefield), this feature provides the following estimated bandwidths: • Module 0 and Module 1: the bandwidth from CPU Module 0 and Module 1 to the DDR • GFX: the bandwidth from the graphics component to the DDR • Display: the bandwidth from the display controller to the DDR • ISP: the bandwidth from the camera image processor to the DDR • IO: the bandwidth between the north and south clusters
`core-temp`	Sampled	IA core temperature statistics, from hardware status data.
`core-volt`	Sampled	Calculate core voltage, from hardware status data. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake.

Name	Collection Methods	Description
`cpu-ddr-mod0-bw` `cpu-ddr-mod1-bw`	Sampled	Precise CPU to DDR module 0 or 1 bandwidth, from hardware signal accumulators. Only one bandwidth or DRAM self-refresh metric can be specified at a time. Hardware counter overflow can occur if -m is not used for signal-based metrics. Supported on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Broxton-M.
`cpu-gpu-concurrency`	Snapshot	Concurrent active time of CPU and GPU, from hardware accumulators.
`ddr-bw`	Sampled	Total DDR memory bandwidth, from hardware accumulators or from signal accumulators. On platforms where signals are used, only one bandwidth or DRAM self-refresh metric can be specified at a time. The hardware accumulator data is always collected over time due to frequent overflow, so snapshot is not available. Hardware counter overflow can occur if -m is not used for signal-based metrics.
`disp-ddr-bw`	Sampled	Precise display controller to DDR memory bandwidth, from hardware signal accumulators. Only one bandwidth or DRAM self-refresh metric can be specified at a time. Hardware counter overflow can occur if `-m` is not used for signal-based metrics. Supported on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Broxton-M.
`dram-pwr`	Sampled	Total DRAM power consumption from hardware accumulators. This data can be collected on Intel Atom Processor-based SoCs for systems code named Apollo Lake and Denverton, and on Intel systems code named Skylake-Xeon.
`dram-srr`	Sampled	DRAM residency in self-refresh mode, from hardware signals. Only one bandwidth or DRAM self-refresh metric can be specified at a time. Hardware counter overflow can occur if -m is not used for signal-based metrics. Do not specify a polling interval greater than 14 seconds when measuring DRAM self-refresh. Supported on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Broxton-M.
`fpga-bw-read`	Sampled	PCIe Read bandwidth by the FPGA.
`fpga-bw-write`	Sampled	PCIe Write bandwidth by the FPGA.
`fpga-pwr`	Sampled	Power consumed by the FPGA.

Name	Collection Methods	Description
`fpga-temp`	Sampled	Temperature of the FPGA.
`gfx-ddr-bw`	Sampled	Graphics component to DDR bandwidth, from hardware signal accumulators. Only one bandwidth or DRAM self-refresh metric can be specified at a time. **NOTE** Some bandwidth measurements include ReadPartial or WritePartial traffic. The payload (how many bytes were transferred) of a partial transaction can range from 1 to 64 bytes. Therefore, the exact bandwidth cannot be accurately measured. In these cases, the Intel SoC Watch bandwidth results will provide a Total Bandwidth Range. The Total Bandwidth Range describes the minimum and maximum bandwidth that was measured. The actual bandwidth falls within the Total Bandwidth Range. Hardware counter overflow can occur if -m is not used for signal-based metrics. Supported on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Broxton-M.
`igfx-throt-rsn`	Sampled	Reasons for throttling the integrated GPU frequency, from hardware status data.
`hw-cpu-cstate`	Snapshot Snapshot , Trace	CPU C-state (sleep) residencies for package/ module/core, from hardware accumulators, and summary of wakeups that cause IA cores to exit a C-state from trace data. Wakeups are only collected if the `-m` or `--max-detail` switch is specified.
`hw-cpu-pstate`	Sampled	CPU P-state operating frequency residencies, from trace data.
`hw-igfx-cstate`	Snapshot	Integrated graphics processor C-state residency (RC6), from hardware accumulators. Always sampled due to short overflow time period.
`hw-igfx-pstate`	Sampled	Integrated graphics processor P-state operating frequency residencies, from hardware status data.
`ia-throt-rsn`	Sampled	Reasons for throttling the CPU frequency, from hardware status data.
`io-bw`	Sampled	IO bandwidth between the North Cluster (NC) and South Cluster (SC), from hardware signal accumulators. Only one bandwidth or DRAM self-refresh metric can be specified at a time. Hardware counter overflow can occur if -m is not used for signal-based metrics. Supported on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Broxton-M.

Name	Collection Methods	Description
`isp-ddr-bw`	Sampled	ISP (camera image processor) to DDR bandwidth, from hardware signal accumulators. Only one bandwidth or DRAM self-refresh metric can be specified at a time. Hardware counter overflow can occur if -m is not used for signal-based metrics. Supported on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Broxton-M.
`nc-dstate`	Sampled	North Complex (NC) component D0ix approximated state residency, from hardware status data. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Cherry Trail.
`netip-bw`	Sampled	Networking IP block to DDR bandwidth, from hardware signal accumulators. Only transfers of 32 or 64 bytes are measured. Use both the `netip-bw` and `netip-partials-bw` features to measure the full networking IP block to DDR bandwidth. Only one bandwidth or DRAM self-refresh metric can be specified at a time. Hardware counter overflow can occur if -m is not used for signal-based metrics.
`netip-partials-bw`	Sampled	Networking IP block to DDR bandwidth, from hardware signal accumulators. Only partial transfers are measured (partial transfers are non-32byte or non-64byte transactions). Use both the `netip-bw` and `netip-partials-bw` features to measure the full networking IP block to DDR bandwidth. Only one bandwidth or DRAM self-refresh metric can be specified at a time. Hardware counter overflow can occur if -m is not used for signal-based metrics.
`pkg-pwr`	Snapshot	Calculate the entire SoC/Package power consumption, from hardware accumulator.
`pmic-temp`	Sampled	MSIC/PMIC temperature, from sysfs reads.
`ring-throt-rsn`	Sampled	Reasons for throttling the ring clock frequency, from hardware status data.
`sc-dstate`	Snapshot or Sampled (depends on target system)	South Complex (SC) component D0ix state residencies. If sampled, the residency is only an approximation. This data can only be collected on Intel Atom Processor-based SoCs for systems code named Cherry Trail and Anniedale .
`skin-temp`	Sampled	Skin temperature data, from sysfs reads.

Name	Collection Methods	Description
soc-temp	Sampled	SoC temperature data, from hardware status data. This data can only be collected on Intel Atom® Processor-based SoCs for systems code named Cherry Trail.
wakelock	Trace	User and kernel wakelock data, from kernel tracepoints and aplog. Android only.

Feature Group Names

The following features are groupings of the previously described features. These group names can be used to simplify command lines to collect multiple features concurrently. For example, `-f cpu` can replace the `-f cpu-cstate -f cpu-pstate` in a command line.

If a group includes a feature that is not enabled on the target platform, that feature will be ignored and collection continue, as long as there is one feature that can be collected.

All features are not supported on all platforms, a group will only include the supported features. Use the `--help` option on the target platform to see the list of group names and specific features included each group.

Name	Description
cpu	cpu-hw
cpu-hw	Most CPU metrics obtained from hardware data sources.
device	Device state residency metrics.
gfx	All graphics metrics from hardware and OS. gfx-hw
gfx-hw	Most GPU metrics obtained from hardware data sources.
hw-gfx-cstate	hw-igfx-cstate
hw-gfx-pstate	hw-igfx-pstate
power	Power/energy metrics.
sstate	System Sx state metrics.
sys	Broad spectrum of metrics commonly used to get general information about platform power behavior.
temp	Temperature metrics.
throt	Frequency throttling reason metrics.

Viewing Intel SoC Watch Results with Intel® VTune™Amplifier 5

You can analyze Intel SoC Watch data graphically using the Intel® VTune™Amplifier GUI. Intel® VTune™Amplifier provides a dynamic timeline view for interacting with Intel SoC Watch data and provides powerful filtering of data for in-depth analysis of a platform's power management behavior.

For detailed instructions, refer to the Analyze Energy Usage section of the Intel® VTune™Amplifier Help.