Intel® System Health Inspector

Version 2.3

User Guide

April 2023

Contents

Quick Start

Use the Intel® System Health Inspector (svr-info) to check system configuration, performance, and profile metrics for one or more Intel® Xeon® servers running Linux* operating systems.

Download

Download svr-info-<*version*>.tgz from Github Releases.

Inspect

Follow these commands to unpack svr-info-<*version*>.tgz, go to the svr-info folder, and run your first system health inspection:

```
tar zxvf svr-info-2.2.0.tgz
cd svr-info
./svr-info
```

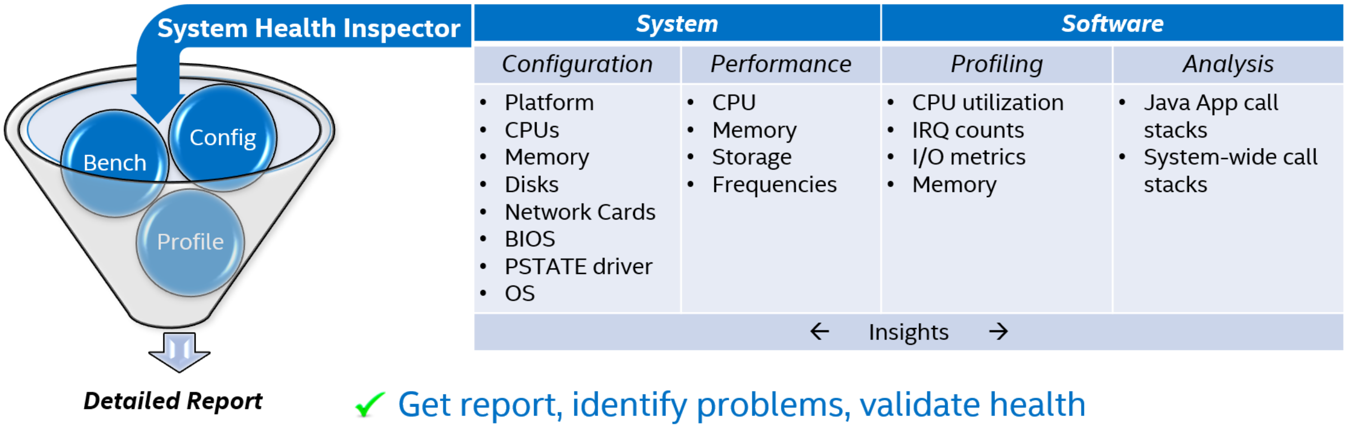
Collect data

Data is limited without elevated privileges.

```
user-mobl          ⢿  finished creating report(s)

Reports:
  svr-info_2022-06-01_18-33-25/user-mobl.html
  svr-info_2022-06-01_18-33-25/user-mobl.xlsx
  svr-info_2022-06-01_18-33-25/user-mobl.json
```

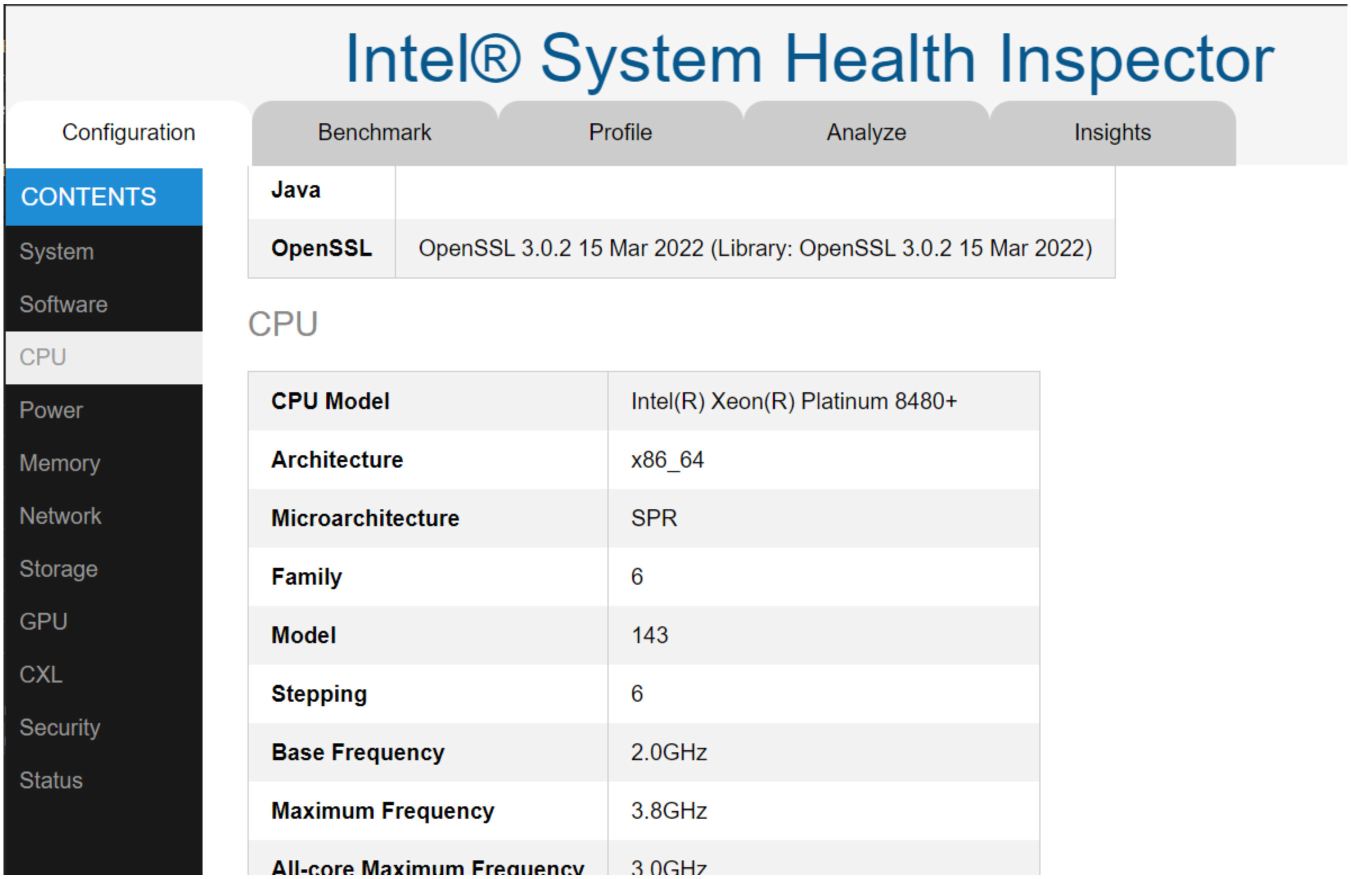
Overview

The Intel® System Health Inspector helps you extract peak performance from your Intel® Xeon based server. It investigates your system profile and tests performance to catch issues and recommend improvements. Inspect one or more nodes. Choose your depth of inspection from quick to detailed. Get reports in HTML, json, or Excel formats. All software dependencies are built-in for a "batteries included" installation.



System		Software	
Configuration	*Performance*	*Profiling*	*Analysis*
• Platform • CPUs • Memory • Disks • Network Cards • BIOS • PSTATE driver • OS	• CPU • Memory • Storage • Frequencies	• CPU utilization • IRQ counts • I/O metrics • Memory	• Java App call stacks • System-wide call stacks
← Insights →			

✓ Get report, identify problems, validate health



Java	
OpenSSL	OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022)

CPU

CPU Model	Intel(R) Xeon(R) Platinum 8480+
Architecture	x86_64
Microarchitecture	SPR
Family	6
Model	143
Stepping	6
Base Frequency	2.0GHz
Maximum Frequency	3.8GHz
All-core Maximum Frequency	3.0GHz

Running the Intel® System Health Inspector

Intel® System Health Inspector has been tested with the CPU architectures shown in the table below. It may run with limited functionality on other architectures. The tar utility must be installed in order to unpack svr-info-<version>.tgz. All dependencies are included.

Architecture	Operating System
Supported Servers	
SPR, SNR, CPX, ICX, CLX, SKX, BDX, HSX	Ubuntu16.04 or newer, Centos7 or newer
Supported Clients	
TGL, RKL, CFL, KBL, SKL, BDW, HSW	Ubuntu16.04 or newer, Centos7 or newer

Note: svr-info may work on other architectures or other Linux distributions, but has not been thoroughly tested.

Follow these commands to unpack svr-info-<version>.tgz. Navigate to the svr-info folder and run your first system health inspection:

```
$ tar zxvf svr-info-2.1.0.tgz
$ cd svr-info
$ ./svr-info
```

The Intel® System Health Inspector supports arguments for gathering data and evaluating your system. Create the types of reports you need, from human-readable management reports to machine-readable data for automated analysis. Add the arguments explained below to the basic svr-info command:

svr-info [-h] [-v]

[-format SELECT]

[-benchmark SELECT] [-storage_dir DIR]

[-profile SELECT] [-profile_duration SECONDS] [-profile_interval N]

[-analyze SELECT] [-analyze_duration SECONDS] [-analyze_frequency N]

[-megadata]

[-ip IP] [-port PORT] [-user USER] [-key KEY] [-targets TARGETS]

[-output OUTPUT] [-temp TEMP] [-dumpconfig] [-cmd_timeout] [-debug]

The following sections explain each parameter and offer usage examples.

Global Arguments

Argument	Description
-h	Intel® System Health Inspector comes with a built in help file that you can access at any time from the command line: `$ ./svr-info -h`

-v	Show the version number of svr-info
	`$ ./svr-info -v`
	`2.2.0`

-output OUTPUT	By default, reports are output to the current folder. To change the output folder, make a new folder or use an existing folder, then specify the path:
	`$ ./svr-info -output /mnt/c/experiments`
	`stacyn1x-mobl          ⣿  finished creating report(s)`
	`Reports:`
	`  experiments/stacyn1x-mobl.html`
	`  experiments/stacyn1x-mobl.xlsx`
	`  experiments/stacyn1x-mobl.json`

-temp TEMP	When svr-info runs, dependencies are extracted into a temporary folder. When the run is complete, this temporary folder is deleted. By default, the temporary folder is the folder assigned to the $TMPDIR environment variable. If $TMPDIR is empty, then /tmp is used. To change the temporary folder, make a new folder or use an existing folder, then specify the path:
	`$ ./svr-info -temp /mnt/c/experiments`
	`stacyn1x-mobl          ⣿  finished creating report(s)`
	`Reports:`

Argument	Description

	`svr-info_2022-06-09_13-04-25/stacyn1x-mobl.html`
	`svr-info_2022-06-09_13-04-25/stacyn1x-mobl.xlsx`
	`svr-info_2022-06-09_13-04-25/stacyn1x-mobl.json`

Report Arguments

Argument	Description

-format FORMAT	By default, html, xlsx, and json report formats are generated, txt is not. The "-format all" option will produce all four formats. To select one or more, add each format in a comma separated list. Report formats include:
	html, xlsx, json, txt, all
	Suppose you want to receive only html and json formats, use this command:
	`$ ./svr-info -format html,json`
	`stacyn1x-mobl          ⣽  finished creating report(s)`
	`Reports:`
	`  svr-info_2022-06-09_12-30-32/stacyn1x-mobl.html`
	`  svr-info_2022-06-09_12-30-32/stacyn1x-mobl.json`
	Note: xlsx denotes the Microsoft Excel format*

Benchmarking Arguments

Choose the depth of inspection by selecting one or more of these performance benchmarking arguments. Micro-benchmarks will run when these arguments are selected. These should only be run when your system is idle.

<table>
<tr><th>Argument</th><th>Description</th></tr>
<tr><td>-benchmark all</td><td>Use the ‘all’ argument to conduct a complete health and performance assessment. Selecting ‘all’ is equivalent to selecting all of these:

cpu,frequency,memory,storage,turbo
<pre>
 $ ./svr-info -benchmark all

stacyn1x-mobl   ⣽ finished creating report(s)

 Reports:
   svr-info_2022-06-09_12-47-18/stacyn1x-mobl.html
   svr-info_2022-06-09_12-47-18/stacyn1x-mobl.xlsx
   svr-info_2022-06-09_12-47-18/stacyn1x-mobl.json
</pre></td></tr>
<tr><td>-benchmark cpu</td><td>Measure CPU performance</td></tr>
<tr><td>-benchmark frequency</td><td>Measure turbo frequencies</td></tr>
<tr><td>-benchmark memory</td><td>Measure memory performance</td></tr>
<tr><td>-benchmark storage</td><td>Measure storage device performance</td></tr>
<tr><td>-storage_dir STORAGE_DIR</td><td>(Optional) The micro-benchmark that measures storage creates a file on the target machine. By default it will be stored in the /tmp folder. Use this argument to specify another folder, possibly on another disk</td></tr>
<tr><td>-benchmark turbo</td><td>Measure TDP and turbo frequencies</td></tr>
</table>

Additional Data Collection Arguments: -profile

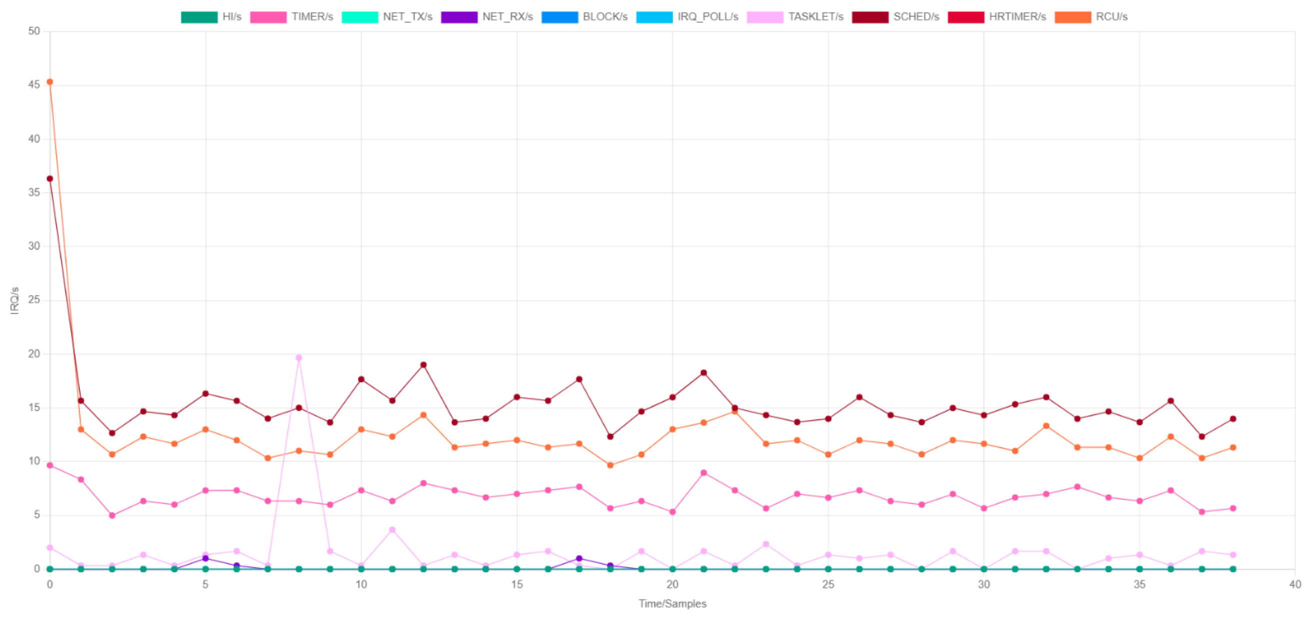
Chart system profile data by inspecting the behavior of the system while it is running a workload. Select the number of samples to be collected along with the time interval in seconds.

Argument	Description
-profile <options>	Options are cpu,network,storage,memory,all.
-profile_interval	Set the number of seconds between each data sample. Default 2.
-profile_duration	Set the number of seconds to collect profiling data. Default 60.

In this example, the IRQ Rate chart shows samples collected at a frequency of 3 seconds.

```
$ ./svr-info -profile all -profile_interval 3 -profile_duration 40
```

IRQ Rate



Additional Data Collection Arguments: -analyze

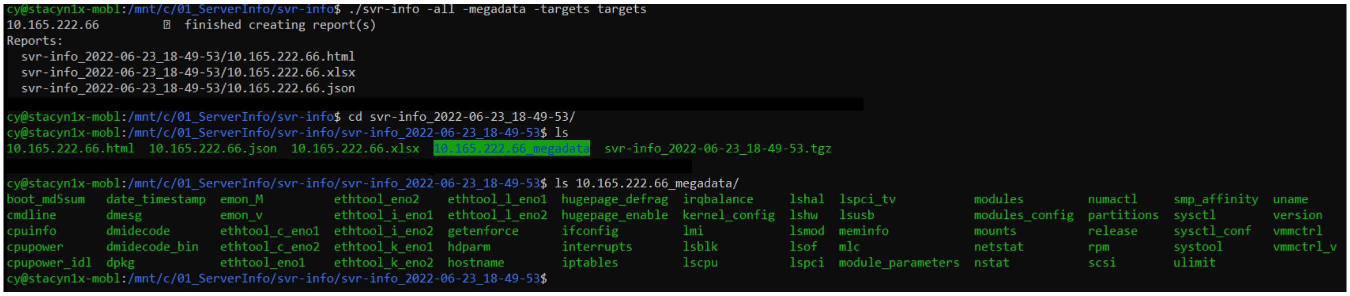
Collect software call stacks to be displayed as Flamegraphs in the HTML report.

Argument	Description
-analyze <options>	Options are system,java,all.
-analyze_frequency	The frequency at which to collect samples. Default 11 ms.
-analyze_duration	Set the number of seconds to collect data. Default 60.

Additional Data Collection Arguments: -megadata

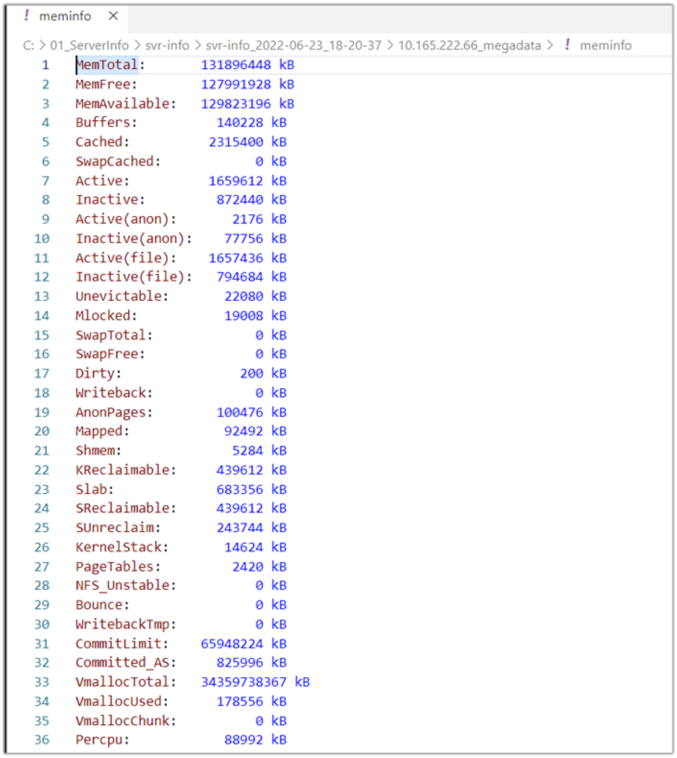
Megadata is an alternative way to collect and present data. You can use the default megadata collection script in the config folder or customize it as needed

Argument	Description
-megadata	Collect the additional data specified in the megadata template file. Ouput individual text files in a folder named *<IP Address>*_megadata.



```
cy@stacyn1x-mobl:/mnt/c/01_ServerInfo/svr-info$ ./svr-info -all -megadata -targets targets
10.165.222.66         □  finished creating report(s)
Reports:
  svr-info_2022-06-23_18-49-53/10.165.222.66.html
  svr-info_2022-06-23_18-49-53/10.165.222.66.xlsx
  svr-info_2022-06-23_18-49-53/10.165.222.66.json

cy@stacyn1x-mobl:/mnt/c/01_ServerInfo/svr-info$ cd svr-info_2022-06-23_18-49-53/
cy@stacyn1x-mobl:/mnt/c/01_ServerInfo/svr-info/svr-info_2022-06-23_18-49-53$ ls
10.165.222.66.html  10.165.222.66.json  10.165.222.66.xlsx  10.165.222.66_megadata  svr-info_2022-06-23_18-49-53.tgz

cy@stacyn1x-mobl:/mnt/c/01_ServerInfo/svr-info/svr-info_2022-06-23_18-49-53$ ls 10.165.222.66_megadata/
boot_md5sum   date_timestamp  emon_M          ethtool_eno2    ethtool_l_eno1  hugepage_defrag  irqbalance     lshal  lspci_tv           modules         numactl     smp_affinity  uname
cmdline       dmesg           emon_v          ethtool_i_eno1  ethtool_l_eno2  hugepage_enable  kernel_config  lshw   lsusb              modules_config  partitions  sysctl        version
cpuinfo       dmidecode       ethtool_c_eno1  ethtool_i_eno2  getenforce      ifconfig         lmi            lsmod  meminfo            mounts          release     sysctl_conf   vmmctrl
cpupower      dmidecode_bin   ethtool_c_eno2  ethtool_k_eno1  hdparm          interrupts       lsblk          lsof   mlc                netstat         rpm         systool       vmmctrl_v
cpupower_idl  dpkg            ethtool_eno1    ethtool_k_eno2  hostname        iptables         lscpu          lspci  module_parameters  nstat           scsi        ulimit
cy@stacyn1x-mobl:/mnt/c/01_ServerInfo/svr-info/svr-info_2022-06-23_18-49-53$
```

Each megadata file contains raw output in text format, for example, **meminfo** contains memory statistics:



C: > 01_ServerInfo > svr-info > svr-info_2022-06-23_18-20-37 > 10.165.222.66_megadata > meminfo

```
MemTotal:       131896448 kB
MemFree:        127991928 kB
MemAvailable:   129823196 kB
Buffers:           140228 kB
Cached:           2315400 kB
SwapCached:             0 kB
Active:           1659612 kB
Inactive:          872440 kB
Active(anon):        2176 kB
Inactive(anon):     77756 kB
Active(file):     1657436 kB
Inactive(file):    794684 kB
Unevictable:        22080 kB
Mlocked:            19008 kB
SwapTotal:              0 kB
SwapFree:               0 kB
Dirty:                200 kB
Writeback:              0 kB
AnonPages:         100476 kB
Mapped:             92492 kB
Shmem:               5284 kB
KReclaimable:      439612 kB
Slab:              683356 kB
SReclaimable:      439612 kB
SUnreclaim:        243744 kB
KernelStack:        14624 kB
PageTables:          2420 kB
NFS_Unstable:           0 kB
Bounce:                 0 kB
WritebackTmp:           0 kB
CommitLimit:     65948224 kB
Committed_AS:      825996 kB
VmallocTotal:   34359738367 kB
VmallocUsed:       178556 kB
VmallocChunk:           0 kB
Percpu:             88992 kB
```

Remote Target Arguments

Intel® System Health Inspector can connect and collect data from one or more remote servers through SSH. To collect data from one server, setup and share your SSH keys then run svr-info from the command line. To collect data from one or more servers, store the connection parameters for each server in the targets file. The targets file may contain either the SSH password or the SSH key.

Argument	Description
-ip IP	Default: localhost. Enter the IP address or hostname for the target server
-port PORT	Default: 22. Enter the SSH port number for the target server
-user USER	Default: Nil. Enter the username used to access the remote target
-key KEY	Default: Nil. Enter the local path to SSH private key file
-targets TARGETS	Default: Nil. Enter the remote connection data necessary to connect to one or more remote servers.

Collecting data from one remote server

Add the IP address, port, username and the local path to the private SSH key. Depending on how you setup your SSH keys, you may be prompted to enter your passphrase.

```
$ ./svr-info -all -ip 10.165.222.169 -port 22 -user sdp -key
/home/username/.ssh/id_rsa

 10.165.222.169        ⡿  finished creating report(s)

Reports:
   svr-info_2022-06-07_13-51-18/10.165.222.169.html
   svr-info_2022-06-07_13-51-18/10.165.222.169.xlsx
   svr-info_2022-06-07_13-51-18/10.165.222.169.json
```

Collecting data from multiple remote servers

Collect data from multiple servers simultaneously by creating a targets file with connection parameters for each remote server. Use one line for each server. Use a colon (:) to separate each argument. List either the private_key_path or the ssh_password. Sudo_password is optional. If not provided, all data will be collected if remote user is configured for password-less sudo. Otherwise, data collected will be limited.

'ip_address:ssh_port:user_name:private_key_path:ssh_password:sudo_password'

In the following example, the first line in the targets file contains this data:

IP address:	10.165.222.169
Port:	:: *(two colons appear because the port is not specified. The default port 22 will be used)*
Username:	username
Private key path:	:: *(no private key path is specified)*
SSH Password:	Passw0rd
Sudo password:	Passw0rd

Optional:

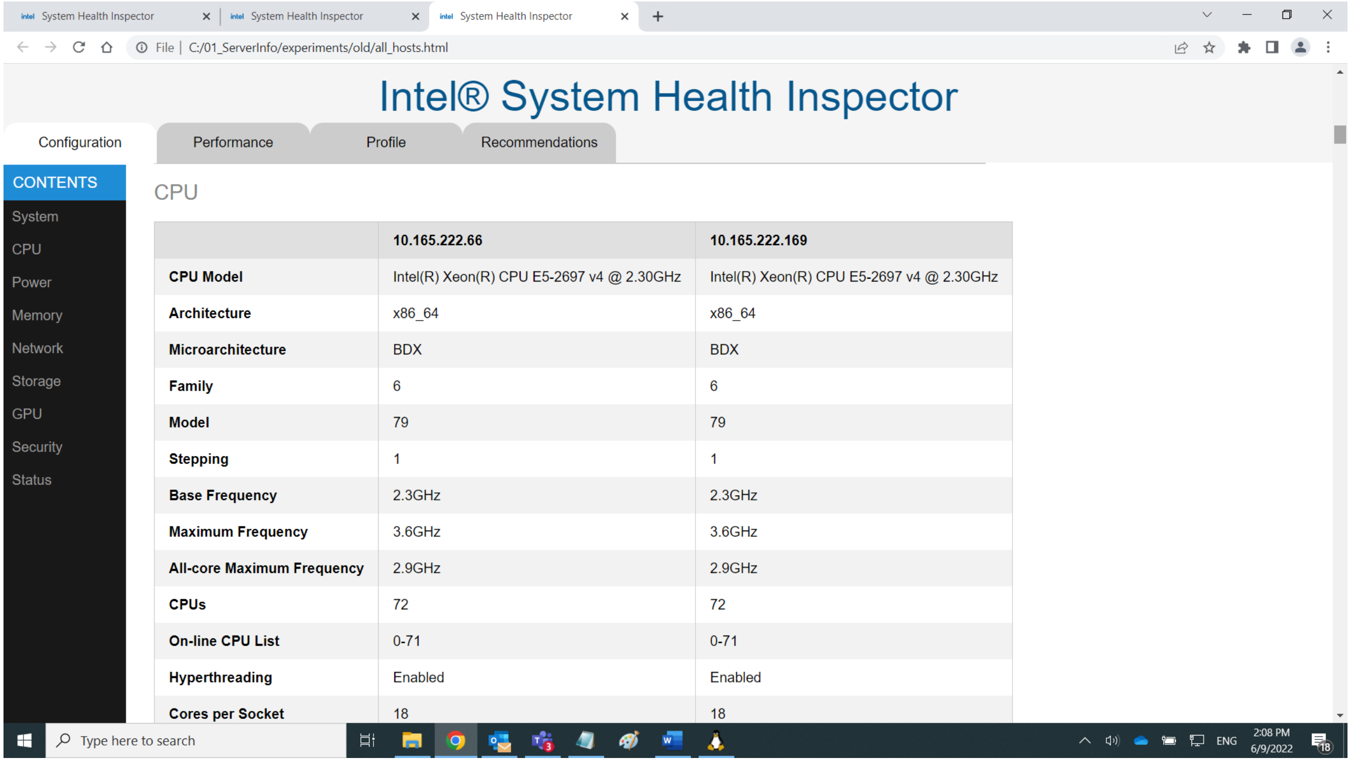
Label	The connection parameters may be proceeded by a label.
Comment	The connections parameters may be followed by a comment.

```
$ cat targets

#ip:port:user:key:pwd:sudo
10.165.222.169::username::Passw0rd:Passw0rd # an optional comment
MY_SERVER:10.165.222.66::username::Passw0rd:Passw0rd # has an optional label

$ ./svr-info -all -targets ./targets

MY_SERVER                   ⣟  finished creating report(s)
10.165.222.66               ⣯  finished creating report(s)

Reports:
  svr-info_2022-06-07_12-24-59/MY_SERVER.html
  svr-info_2022-06-07_12-24-59/10.165.222.169.html
  svr-info_2022-06-07_12-24-59/all_hosts.html
  svr-info_2022-06-07_12-24-59/MY_SERVER.xlsx
  svr-info_2022-06-07_12-24-59/10.165.222.169.xlsx
  svr-info_2022-06-07_12-24-59/all_hosts.xlsx
  svr-info_2022-06-07_12-24-59/MY_SERVER.json
  svr-info_2022-06-07_12-24-59/10.165.222.169.json
```

```
svr-info_2022-06-07_12-24-59/all_hosts.json
```

Output for Multiple Nodes

A health assessment report is created for each remote server along with a full report containing data from all the servers. These are stored in files named “all_hosts” plus the report suffix.



Intel® System Health Inspector

CPU

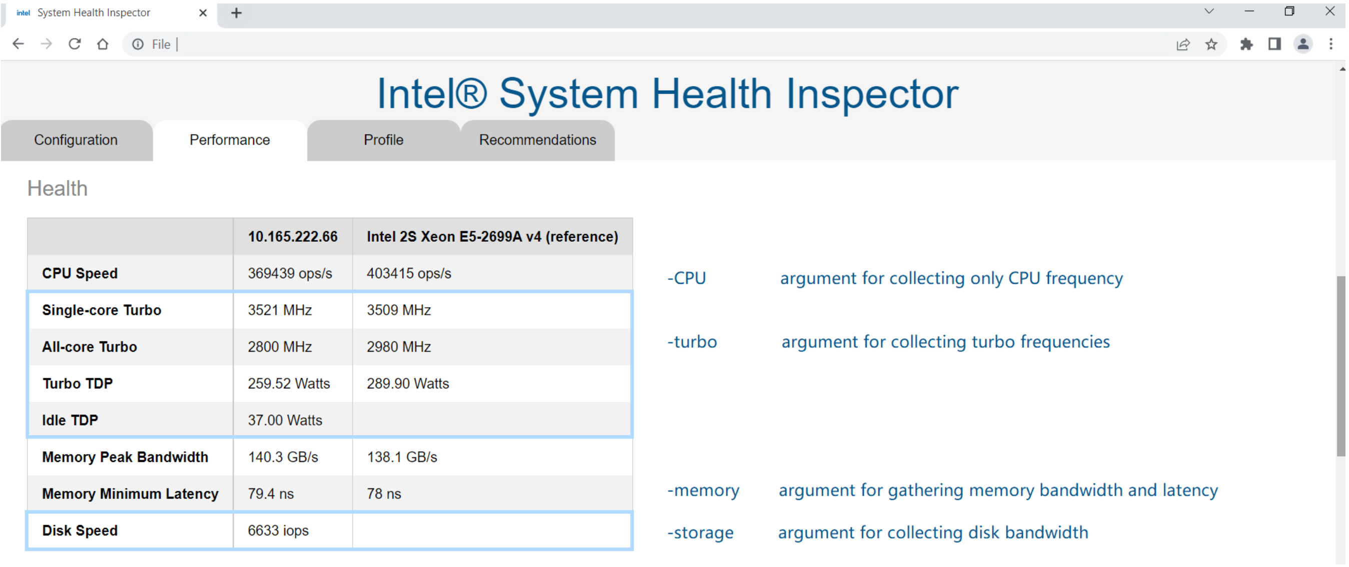
	10.165.222.66	10.165.222.169
CPU Model	Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz	Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz
Architecture	x86_64	x86_64
Microarchitecture	BDX	BDX
Family	6	6
Model	79	79
Stepping	1	1
Base Frequency	2.3GHz	2.3GHz
Maximum Frequency	3.6GHz	3.6GHz
All-core Maximum Frequency	2.9GHz	2.9GHz
CPUs	72	72
On-line CPU List	0-71	0-71
Hyperthreading	Enabled	Enabled
Cores per Socket	18	18

Micro-benchmarks

Micro-benchmarks are used to assess the health of a system. They should be run when the system is idle. Running a micro-benchmark on a production system can cause applications to perform badly because resources are being taken to run the benchmark.

Micro-benchmarks measure memory bandwidth and latency, CPU frequency, turbo frequencies, and disk bandwith. To run all the micro-benchmarks, use the -benchmark all command line argument:

```
$ ./svr-info -benchmark all
```



Health

	10.165.222.66	Intel 2S Xeon E5-2699A v4 (reference)
CPU Speed	369439 ops/s	403415 ops/s
Single-core Turbo	3521 MHz	3509 MHz
All-core Turbo	2800 MHz	2980 MHz
Turbo TDP	259.52 Watts	289.90 Watts
Idle TDP	37.00 Watts	
Memory Peak Bandwidth	140.3 GB/s	138.1 GB/s
Memory Minimum Latency	79.4 ns	78 ns
Disk Speed	6633 iops	

-CPU argument for collecting only CPU frequency

-turbo argument for collecting turbo frequencies

-memory argument for gathering memory bandwidth and latency

-storage argument for collecting disk bandwidth

Use the -CPU option to run a micro-benchmark that assesses the processing speed of the CPU.

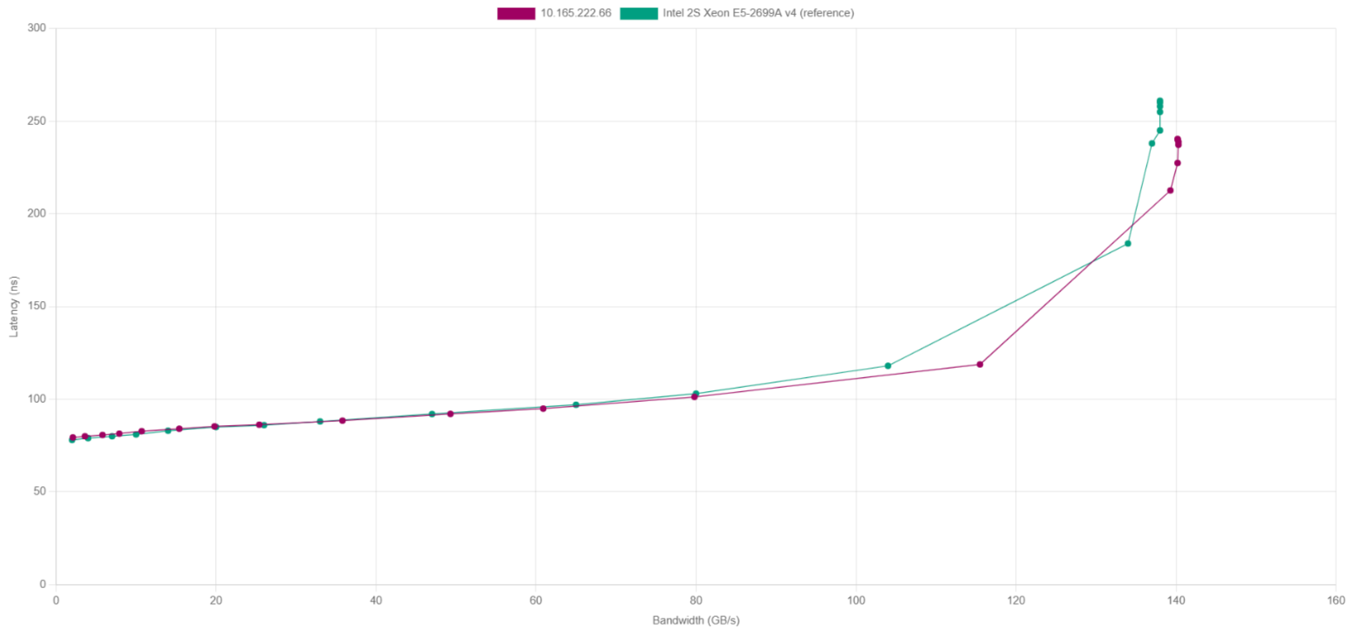
Use the -turbo option to run micro-benchmarks that measure turbo frequencies and TDP.

Use the -memory option to view the peak memory bandwidth and the optimum latency.

Use the -storage to show disk bandwidth

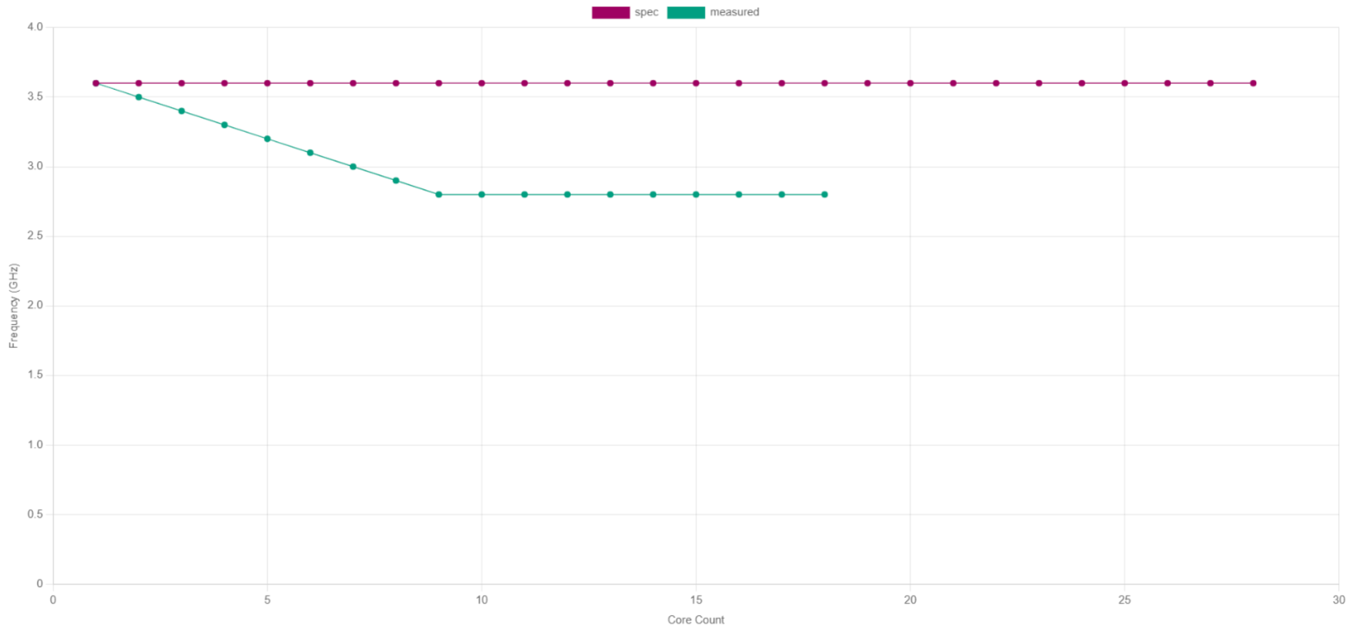
Micro-benchmark: Memory Bandwidth and Latency

Memory latency and bandwidth are measured when the -memory argument is present. The red line shows the mapping from your node. The green line shows the Intel® Xeon® reference mapping.



Micro-benchmark: Core Frequency

Maximum frequency for increasing core counts is measured when the -frequency argument is present. The red line shows the CPU's frequency specification, and the green line shows the measured frequencies.



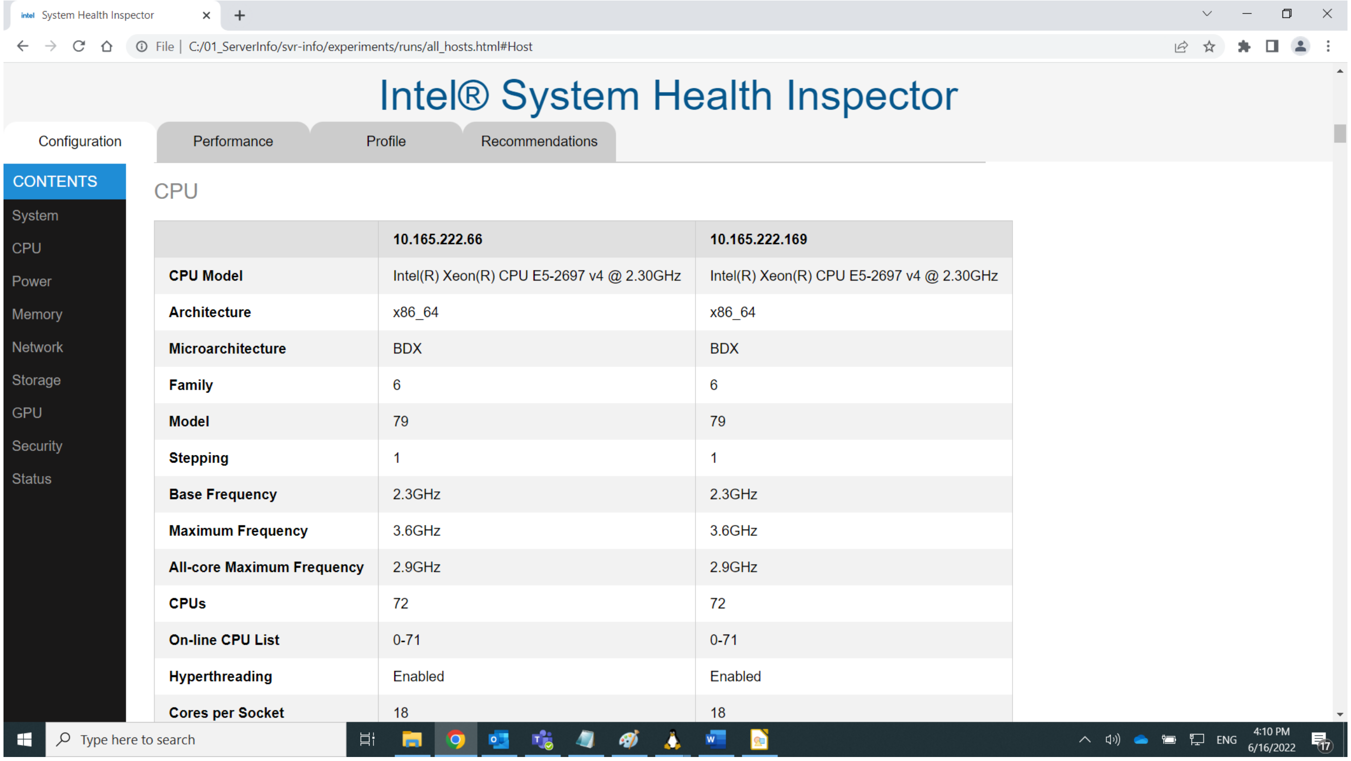
Report Formats

The Intel® System Health Inspector outputs a health status report in three formats: html, xlsx, and json. You can also output a text only version. Report files are named with either the local machine name or the IP address. By default, each set of reports is saved in a folder with a unique name. When you use the -output argument, take care to change the output folder name every time so you do not inadvertently overwrite your reports.

```
$ ./svr-info

stacyn1x-mobl          ⠿  finished creating report(s)

Reports:
  svr-info_2022-06-06_16-58-54/stacyn1x-mobl.html
  svr-info_2022-06-06_16-58-54/stacyn1x-mobl.xlsx
  svr-info_2022-06-06_16-58-54/stacyn1x-mobl.json
```

HTML

HTML reports contain categories of data that are shown in HTML as tabs: Configuration, Performance, Profile, and Recommendations.



CPU

	10.165.222.66	10.165.222.169
CPU Model	Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz	Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz
Architecture	x86_64	x86_64
Microarchitecture	BDX	BDX
Family	6	6
Model	79	79
Stepping	1	1
Base Frequency	2.3GHz	2.3GHz
Maximum Frequency	3.6GHz	3.6GHz
All-core Maximum Frequency	2.9GHz	2.9GHz
CPUs	72	72
On-line CPU List	0-71	0-71
Hyperthreading	Enabled	Enabled
Cores per Socket	18	18

Figure 1: Intel® System Health Inspector Report in HTML format

Configuration tab

The configuration tab includes information about the platform, CPU, software installed, etc.

Benchmark tab

The benchmark tab shows micro-benchmark results, if collected.

Profile tab

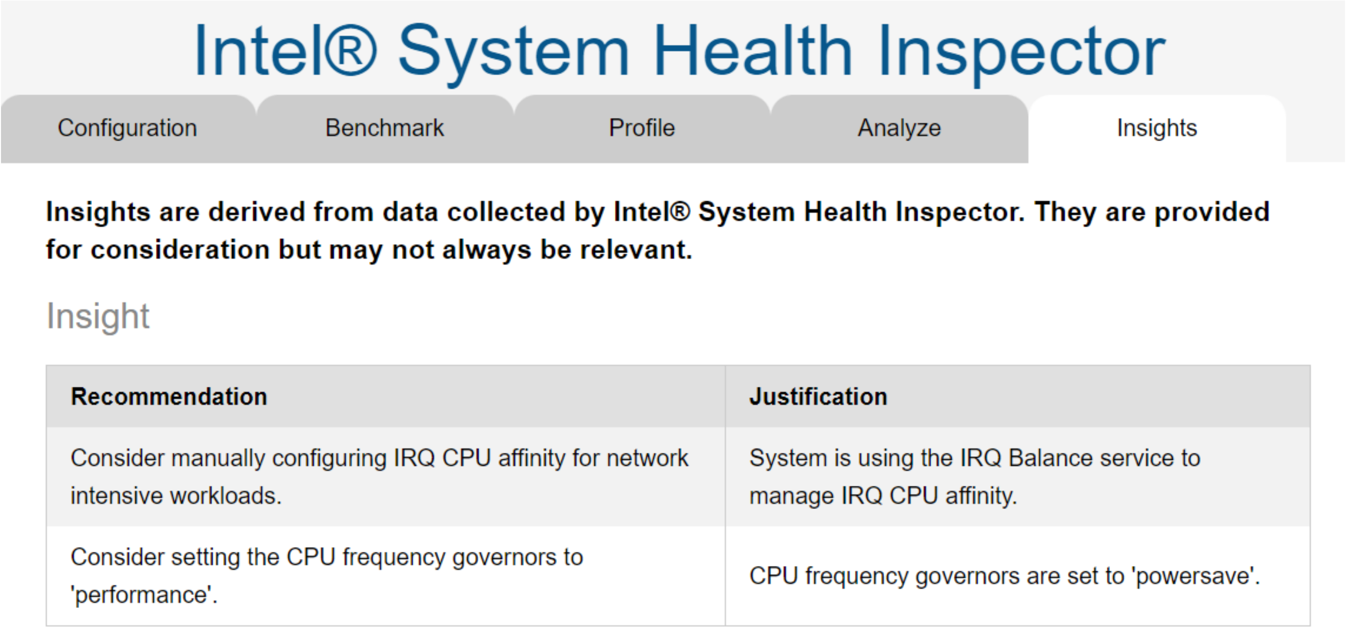
The profile tab shows system profiling telemetry, if collected.

Analyze tab

The analyze tab shows software call stacks, if collected.

Insights tab

The insights tab shows recommendations based on data collected.



Insights are derived from data collected by Intel® System Health Inspector. They are provided for consideration but may not always be relevant.

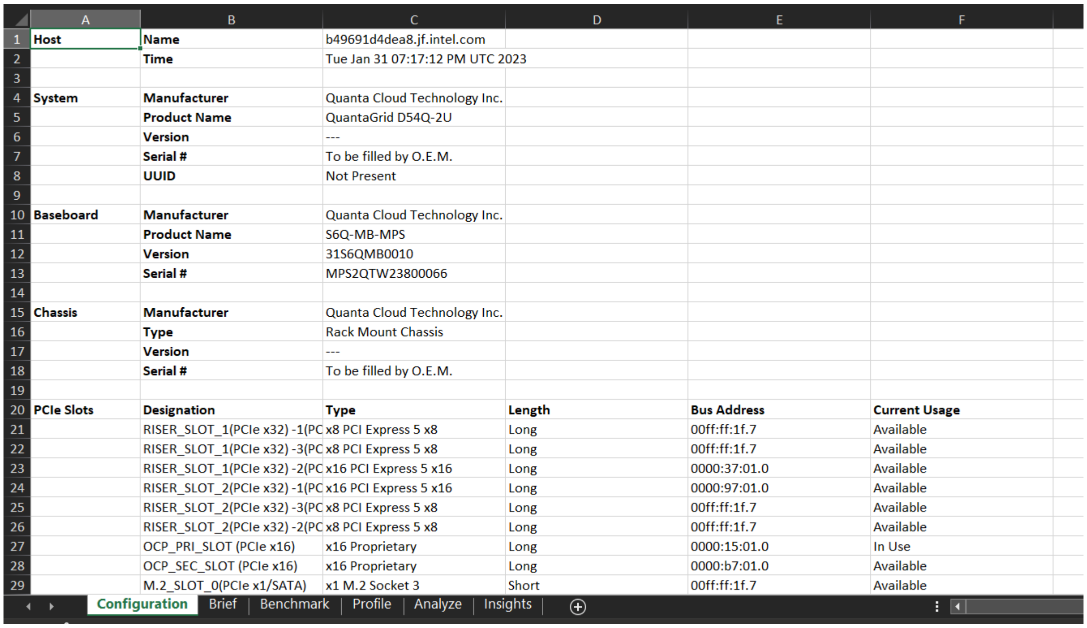
Insight

Recommendation	Justification
Consider manually configuring IRQ CPU affinity for network intensive workloads.	System is using the IRQ Balance service to manage IRQ CPU affinity.
Consider setting the CPU frequency governors to 'performance'.	CPU frequency governors are set to 'powersave'.

Microsoft Excel* (xlsx)

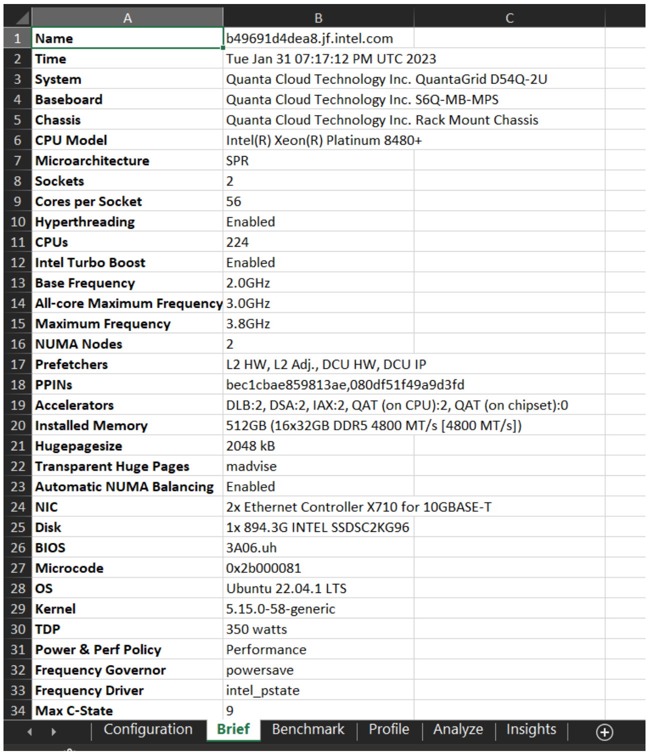
Excel* shows system health data in six worksheets: Configuration, Brief, Benchmark, Profile, Analyze, and Insights.

The Configuration worksheet contains one column for each system.



	A	B	C	D	E	F
1	**Host**	**Name**	b49691d4dea8.jf.intel.com			
2		**Time**	Tue Jan 31 07:17:12 PM UTC 2023			
3						
4	**System**	**Manufacturer**	Quanta Cloud Technology Inc.			
5		**Product Name**	QuantaGrid D54Q-2U			
6		**Version**	---			
7		**Serial #**	To be filled by O.E.M.			
8		**UUID**	Not Present			
9						
10	**Baseboard**	**Manufacturer**	Quanta Cloud Technology Inc.			
11		**Product Name**	S6Q-MB-MPS			
12		**Version**	31S6QMB0010			
13		**Serial #**	MPS2QTW23800066			
14						
15	**Chassis**	**Manufacturer**	Quanta Cloud Technology Inc.			
16		**Type**	Rack Mount Chassis			
17		**Version**	---			
18		**Serial #**	To be filled by O.E.M.			
19						
20	**PCIe Slots**	**Designation**	**Type**	**Length**	**Bus Address**	**Current Usage**
21		RISER_SLOT_1(PCIe x32) -1(PC	x8 PCI Express 5 x8	Long	00ff:ff:1f.7	Available
22		RISER_SLOT_1(PCIe x32) -3(PC	x8 PCI Express 5 x8	Long	00ff:ff:1f.7	Available
23		RISER_SLOT_1(PCIe x32) -2(PC	x16 PCI Express 5 x16	Long	0000:37:01.0	Available
24		RISER_SLOT_2(PCIe x32) -1(PC	x16 PCI Express 5 x16	Long	0000:97:01.0	Available
25		RISER_SLOT_2(PCIe x32) -3(PC	x8 PCI Express 5 x8	Long	00ff:ff:1f.7	Available
26		RISER_SLOT_2(PCIe x32) -2(PC	x8 PCI Express 5 x8	Long	00ff:ff:1f.7	Available
27		OCP_PRI_SLOT (PCIe x16)	x16 Proprietary	Long	0000:15:01.0	In Use
28		OCP_SEC_SLOT (PCIe x16)	x16 Proprietary	Long	0000:b7:01.0	Available
29		M.2_SLOT_0(PCIe x1/SATA)	x1 M.2 Socket 3	Short	00ff:ff:1f.7	Available

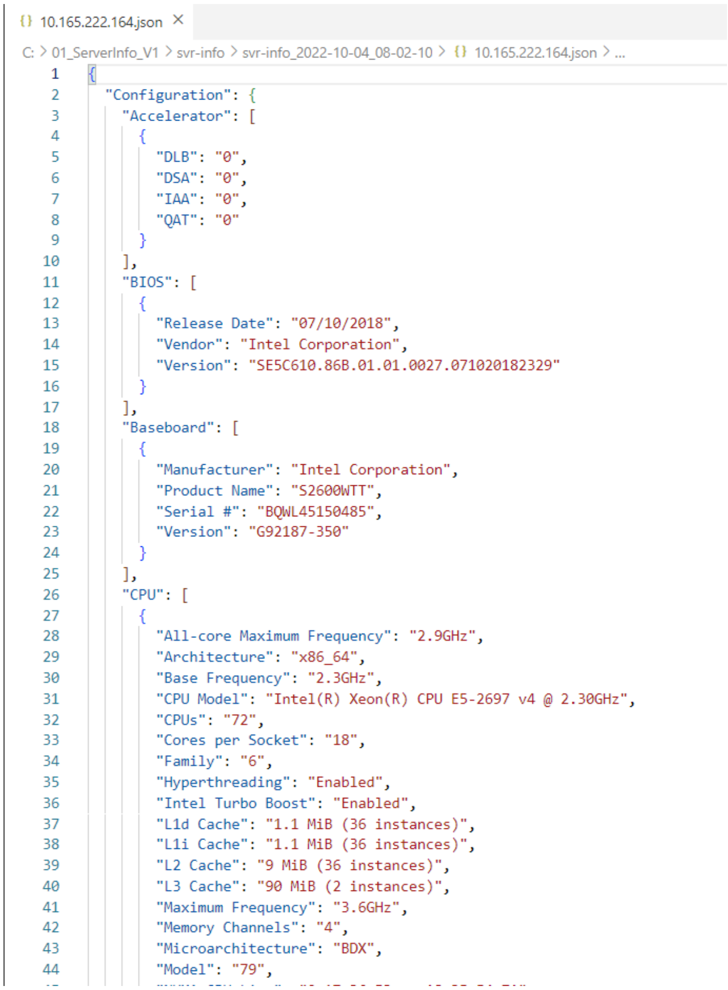
The Brief worksheet provides a summary view useful for copy/paste into presentations and reports.



	A	B	C
1	**Name**	b49691d4dea8.jf.intel.com	
2	**Time**	Tue Jan 31 07:17:12 PM UTC 2023	
3	**System**	Quanta Cloud Technology Inc. QuantaGrid D54Q-2U	
4	**Baseboard**	Quanta Cloud Technology Inc. S6Q-MB-MPS	
5	**Chassis**	Quanta Cloud Technology Inc. Rack Mount Chassis	
6	**CPU Model**	Intel(R) Xeon(R) Platinum 8480+	
7	**Microarchitecture**	SPR	
8	**Sockets**	2	
9	**Cores per Socket**	56	
10	**Hyperthreading**	Enabled	
11	**CPUs**	224	
12	**Intel Turbo Boost**	Enabled	
13	**Base Frequency**	2.0GHz	
14	**All-core Maximum Frequency**	3.0GHz	
15	**Maximum Frequency**	3.8GHz	
16	**NUMA Nodes**	2	
17	**Prefetchers**	L2 HW, L2 Adj., DCU HW, DCU IP	
18	**PPINs**	bec1cbae859813ae,080df51f49a9d3fd	
19	**Accelerators**	DLB:2, DSA:2, IAX:2, QAT (on CPU):2, QAT (on chipset):0	
20	**Installed Memory**	512GB (16x32GB DDR5 4800 MT/s [4800 MT/s])	
21	**Hugepagesize**	2048 kB	
22	**Transparent Huge Pages**	madvise	
23	**Automatic NUMA Balancing**	Enabled	
24	**NIC**	2x Ethernet Controller X710 for 10GBASE-T	
25	**Disk**	1x 894.3G INTEL SSDSC2KG96	
26	**BIOS**	3A06.uh	
27	**Microcode**	0x2b000081	
28	**OS**	Ubuntu 22.04.1 LTS	
29	**Kernel**	5.15.0-58-generic	
30	**TDP**	350 watts	
31	**Power & Perf Policy**	Performance	
32	**Frequency Governor**	powersave	
33	**Frequency Driver**	intel_pstate	
34	**Max C-State**	9	

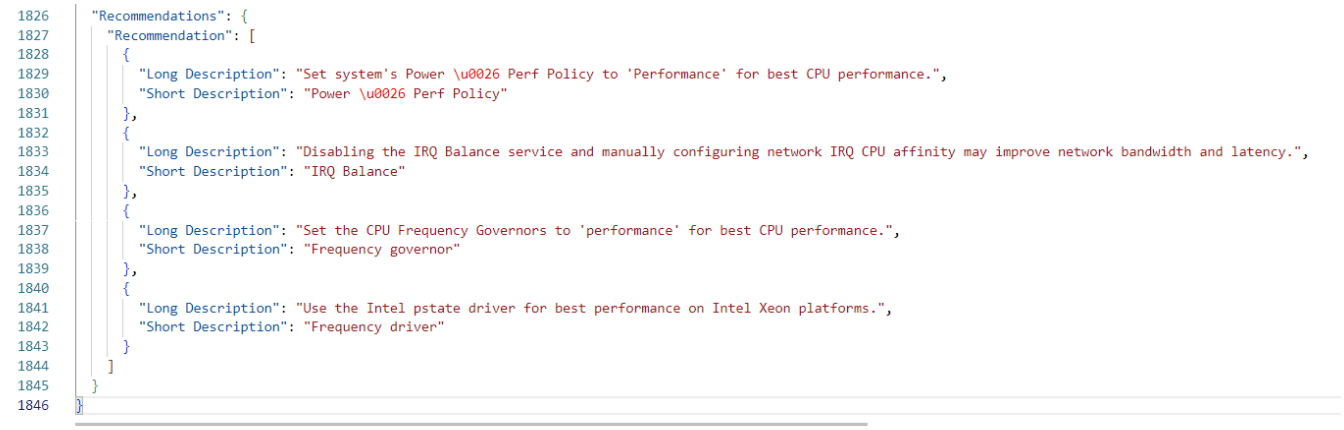
JSON

The json schema contains the hostname and the data collected for each host. Each command that was run to gather data has a name and category along with the type of data gathered and the actual data. In this example, BIOS data is collected including the release date, vendor, and version. Baseboard data includes Manufacturer, Product Name, Serial #, and Version. The final example shows the recommendations.


{} 10.165.222.164.json ×
C: > 01_ServerInfo_V1 > svr-info > svr-info_2022-10-04_08-02-10 > {} 10.165.222.164.json > ...

```
{
  "Configuration": {
    "Accelerator": [
      {
        "DLB": "0",
        "DSA": "0",
        "IAA": "0",
        "QAT": "0"
      }
    ],
    "BIOS": [
      {
        "Release Date": "07/10/2018",
        "Vendor": "Intel Corporation",
        "Version": "SE5C610.86B.01.01.0027.071020182329"
      }
    ],
    "Baseboard": [
      {
        "Manufacturer": "Intel Corporation",
        "Product Name": "S2600WTT",
        "Serial #": "BQWL45150485",
        "Version": "G92187-350"
      }
    ],
    "CPU": [
      {
        "All-core Maximum Frequency": "2.9GHz",
        "Architecture": "x86_64",
        "Base Frequency": "2.3GHz",
        "CPU Model": "Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz",
        "CPUs": "72",
        "Cores per Socket": "18",
        "Family": "6",
        "Hyperthreading": "Enabled",
        "Intel Turbo Boost": "Enabled",
        "L1d Cache": "1.1 MiB (36 instances)",
        "L1i Cache": "1.1 MiB (36 instances)",
        "L2 Cache": "9 MiB (36 instances)",
        "L3 Cache": "90 MiB (2 instances)",
        "Maximum Frequency": "3.6GHz",
        "Memory Channels": "4",
        "Microarchitecture": "BDX",
        "Model": "79",
```

View the entire json file for each run in the folder named svr-info_*timestamp*.



```
  "Recommendations": {
    "Recommendation": [
      {
        "Long Description": "Set system's Power \u0026 Perf Policy to 'Performance' for best CPU performance.",
        "Short Description": "Power \u0026 Perf Policy"
      },
      {
        "Long Description": "Disabling the IRQ Balance service and manually configuring network IRQ CPU affinity may improve network bandwidth and latency.",
        "Short Description": "IRQ Balance"
      },
      {
        "Long Description": "Set the CPU Frequency Governors to 'performance' for best CPU performance.",
        "Short Description": "Frequency governor"
      },
      {
        "Long Description": "Use the Intel pstate driver for best performance on Intel Xeon platforms.",
        "Short Description": "Frequency driver"
      }
    ]
  }
}
```

Using the Reporter independently

Suppose you initially ran the reports in HTML format and now you need reports in json format, too. The reporter makes it easy to run reports from a prior collection in one or more report formats. Reporter is included within the svr-info binary. Access it using the -reporter command line option. Add the global arguments to the basic reporter command to select the format, the raw input file(s) and your output folder.

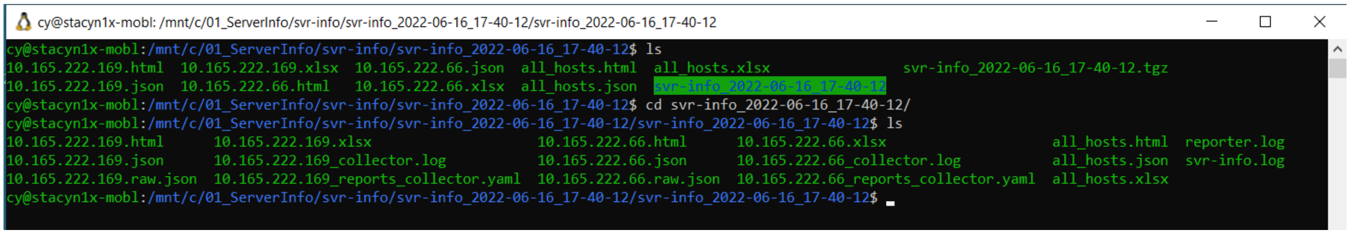
svr-info -reporter "[-format FORMAT] [-input *.raw.json] [-output OUTPUT]"

Reporter Global Arguments

Argument	Description
-h	The reporter comes with a built in help file that you can access at any time from the command line. Navigate to the /tools folder and run: `$ ./svr-info -reporter "-h"`
-format	By default, all report formats will be output. To select one or more, add each format in a comma separated list. Report formats include these: html, xlsx, json, all
-input INPUT	Add a comma separated list of input files or a folder containing the input files. Input files follow this naming convention: *.raw.json
-output OUTPUT	By default, reports are output to the current folder. To change the output folder, make a new folder or use an existing folder, then specify the path

Finding raw health inspection data

The Intel® System Health Inspector stores raw data from each inspection in a .tgz file with the same date and time stamp as the report folder, for example: svr-info_2022-06-16_17-40-12.tgz. Navigate to the report folder and unzip the tgz file to find the *.raw.json file(s).



cy@stacyn1x-mobl: /mnt/c/01_ServerInfo/svr-info/svr-info_2022-06-16_17-40-12/svr-info_2022-06-16_17-40-12

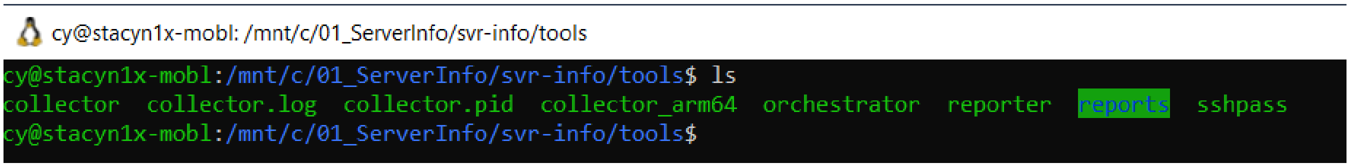
```
cy@stacyn1x-mobl:/mnt/c/01_ServerInfo/svr-info/svr-info_2022-06-16_17-40-12$ ls
10.165.222.169.html  10.165.222.169.xlsx  10.165.222.66.json  all_hosts.html  all_hosts.xlsx                svr-info_2022-06-16_17-40-12.tgz
10.165.222.169.json  10.165.222.66.html   10.165.222.66.xlsx  all_hosts.json  svr-info_2022-06-16_17-40-12
cy@stacyn1x-mobl:/mnt/c/01_ServerInfo/svr-info/svr-info_2022-06-16_17-40-12$ cd svr-info_2022-06-16_17-40-12/
cy@stacyn1x-mobl:/mnt/c/01_ServerInfo/svr-info/svr-info_2022-06-16_17-40-12/svr-info_2022-06-16_17-40-12$ ls
10.165.222.169.html      10.165.222.169.xlsx                    10.165.222.66.html      10.165.222.66.xlsx                    all_hosts.html  reporter.log
10.165.222.169.json      10.165.222.169_collector.log           10.165.222.66.json      10.165.222.66_collector.log           all_hosts.json  svr-info.log
10.165.222.169.raw.json  10.165.222.169_reports_collector.yaml  10.165.222.66.raw.json  10.165.222.66_reports_collector.yaml  all_hosts.xlsx
cy@stacyn1x-mobl:/mnt/c/01_ServerInfo/svr-info/svr-info_2022-06-16_17-40-12/svr-info_2022-06-16_17-40-12$
```

Contents of Intel® System Health Inspector raw data tgz file:

```
10.165.222.169.html
10.165.222.169.json
10.165.222.169.raw.json
10.165.222.169.xlsx
10.165.222.169_collector.log
10.165.222.169_reports_collector.yaml
10.165.222.66.html
10.165.222.66.json
10.165.222.66.raw.json
10.165.222.66.xlsx
10.165.222.66_collector.log
10.165.222.66_reports_collector.yaml
all_hosts.html
all_hosts.json
all_hosts.xlsx
reporter.log
svr-info.log
```

Running a new report in JSON format

1. Navigate to the svr-info/tools directory and create a new folder called /reports.



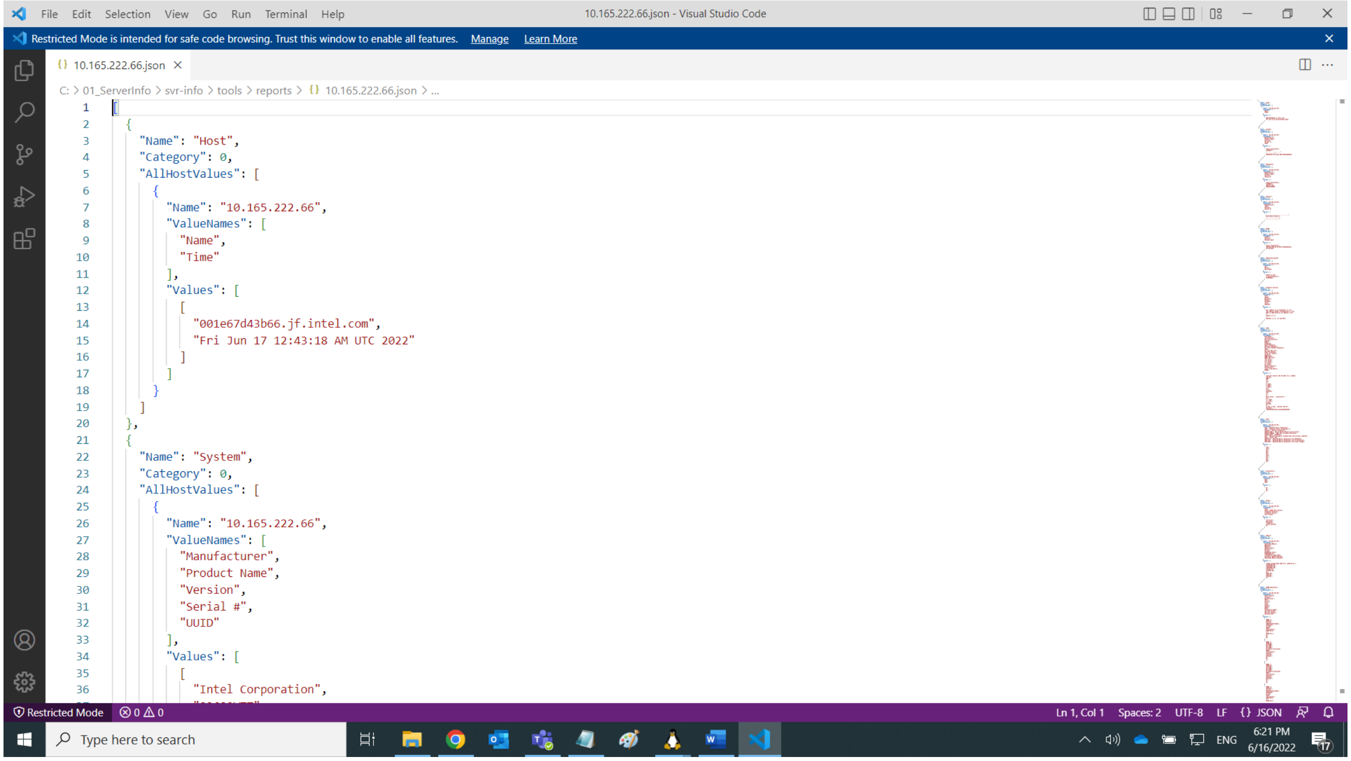
cy@stacyn1x-mobl: /mnt/c/01_ServerInfo/svr-info/tools

```
cy@stacyn1x-mobl:/mnt/c/01_ServerInfo/svr-info/tools$ ls
collector  collector.log  collector.pid  collector_arm64  orchestrator  reporter  reports  sshpass
cy@stacyn1x-mobl:/mnt/c/01_ServerInfo/svr-info/tools$
```

2. Enter the ./reporter command and select the json format. Add the path to the input. Set the output folder to reports. Run the reporter.

```
$ ./reporter -format json -input /mnt/c/01_ServerInfo/svr-info/svr-info_2022-
  06-16_17-40-12/svr-info_2022-06-16_17-40-12/10.165.222.66.raw.json -output
  reports

  reports/10.165.222.66.json
```

3. Navigate to the reports folder to find your new report in json format. The following screen capture identifies the host IP address and the name plus the time the report was run.



Restricted Mode is intended for safe code browsing. Trust this window to enable all features.

```
[
  {
    "Name": "Host",
    "Category": 0,
    "AllHostValues": [
      {
        "Name": "10.165.222.66",
        "ValueNames": [
          "Name",
          "Time"
        ],
        "Values": [
          [
            "001e67d43b66.jf.intel.com",
            "Fri Jun 17 12:43:18 AM UTC 2022"
          ]
        ]
      }
    ]
  },
  {
    "Name": "System",
    "Category": 0,
    "AllHostValues": [
      {
        "Name": "10.165.222.66",
        "ValueNames": [
          "Manufacturer",
          "Product Name",
          "Version",
          "Serial #",
          "UUID"
        ],
        "Values": [
          [
            "Intel Corporation",
```

Using the Collector independently

Suppose you are responsible for maintaining multiple servers in a datacenter. You need to collect system health data from each server as input into another system. Use the collector to gather system health data. The collector is included within the svr-info binary. To access the collector use the svr-info -collector command line option

Collector Global Arguments

Argument	Description
-h	The collector comes with a built in help file that you can access at any time from the command line. Navigate to the /tools folder and run: `$ ./svr-info -collector “-h”`

-v	Show the version number of the collector
	`$ ./svr-info -collector “-v”`
	`2.0.1_2022-05-06_053c29ea`

YAML File Format

The YAML format contains both the commands and arguments for the root level keys. Commands used to gather data will be executed by bash. Each command must have a name that is used to label its output. Commands may have required and/or optional arguments . Attributes include:

- **superuser** is a boolean indicating whether to run the command with elevated privileges. The default is false. The collector will read the password from the environment variable SUDO_PASSWORD, if provided.

- **run** is a boolean indicating whether to run the command or not. Often the YAML file will include all available commands. You may set the flag to false if you do not want to collect the data.

- **modprobe** is a comma separated list of kernel modules required to run a command.

- **idle** is a boolean indicating whether the system CPU idle percentage should be collected before running the command. The default is false.

- **parallel** is a boolean indicating whether the command can be run in parallel with other commands. The default is false.

The collector and the YAML file are copied onto the target at runtime. Data collected is stored on the on target in the /tmp folder unless you specify a different folder. $NIC is the environment variable that will be called by commands gathering data from physical network interfaces.

YAML Example

Try out the collector by using this simple example. Under arguments, the name is the name for the root key in the json output. In this example, it is “cy_example”. The bin_path is set to “.” (stdout). Do not use tabs for formatting, YAML format requires spaces.

```
############
# global arguments
############
arguments:
    name:  cy_example
    bin_path: .
############
# commands --
############
commands:
- MAC Address $NIC:
    command: cat /sys/class/net/$NIC/address
    parallel: true
- date -u:
    command: date -u
    parallel: true
- cpuid -1:
    command: cpuid -1 | grep family
    modprobe: cpuid
    parallel: true
```

To run the collector, copy the YAML example into a file. Provide the path to the YAML file as an argument for the collector. Results will be printed to stdout in JSON format. When the collector runs, it also generates a collector.log file and a collector.pid file.

```
$ ./collector /mnt/c/01_ServerInfo/svr-info/config/example.yaml.tmpl
{
"cy_example": [
{
  "command": "date -u",
  "exitstatus": "0",
  "label": "date -u",
  "stderr": "",
  "stdout": "Fri Jun 17 19:20:37 UTC 2022\n",
  "superuser": "false"
}
,{
  "command": "cpuid -1 | grep family",
  "exitstatus": "1",
  "label": "cpuid -1",
  "stderr": "bash: cpuid: command not found\n",
  "stdout": "",
  "superuser": "false"
}
]
}
```

Advanced Topics

This section contains additional information.

Inspecting the Public Cloud

When the remote server is in the Cloud, make sure the Cloud instance has a public IP address and the inbound firewall rules allow access via port 22. Cloud instances will not report data withheld by the hypervisor and unavailable to the guest operating system.

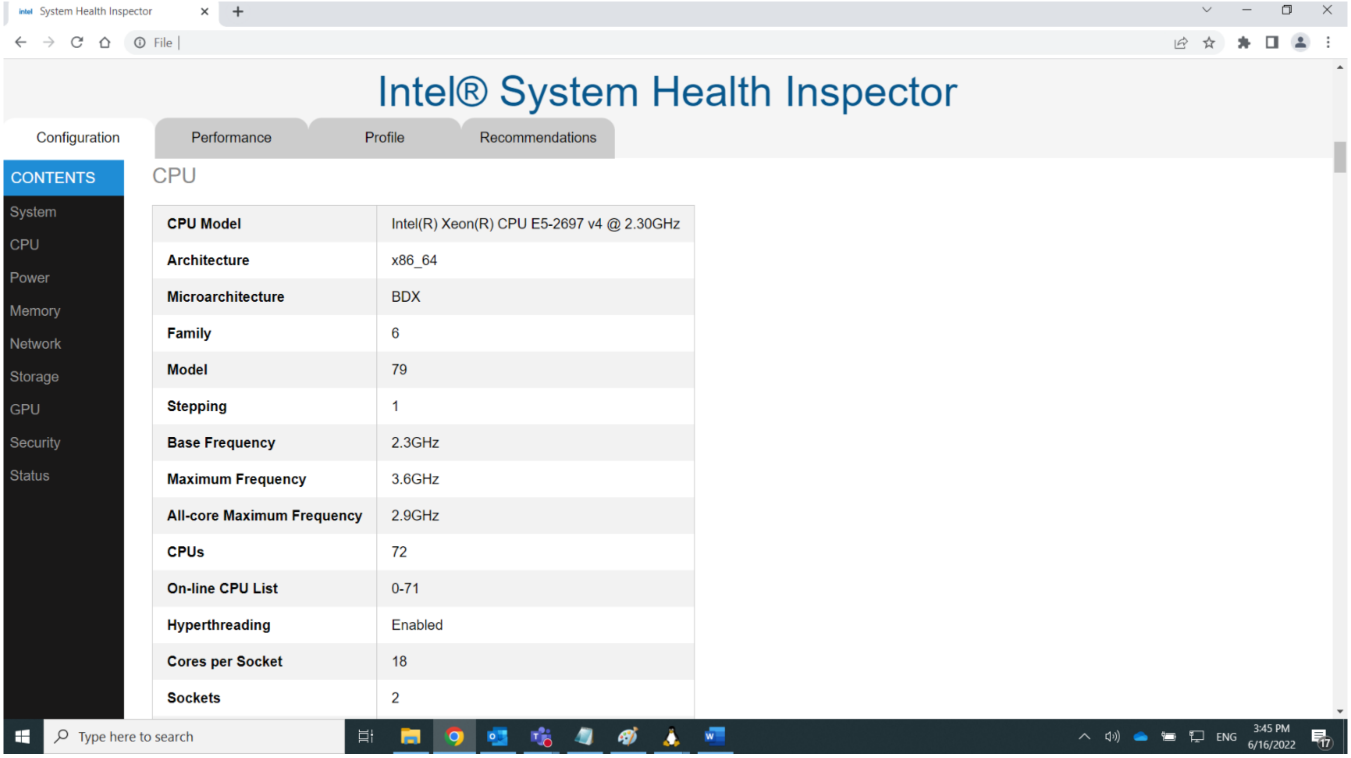
Measuring Frequencies

The CPU table in the HTML report has two frequency-related fields: Maximum Frequency and All-Core Maximum Frequency. Maximum Frequency is determined by first looking for the cpuinfo_max_freq file:

/sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_max_freq

If this file does not exist, then data is gathered from MSR 0x1ad. If that fails, the frequency is read from the "Max Speed" field in dmidecode's output.

All-Core Maximum Frequency is read from MSR 0x1ad.

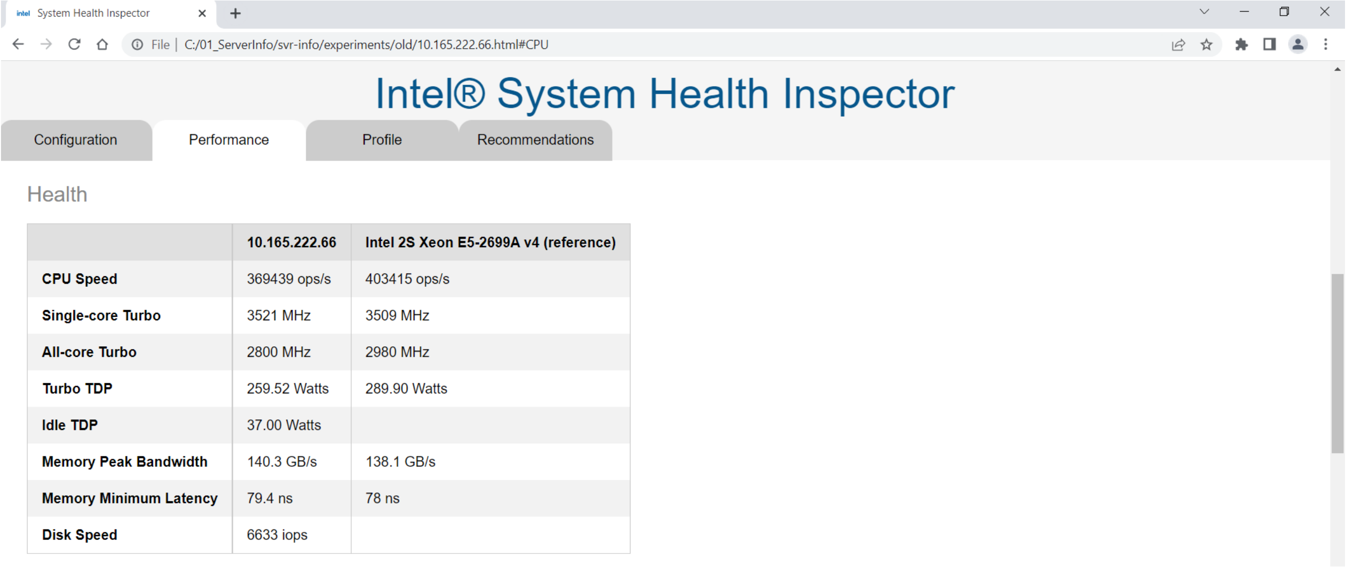


CPU

CPU Model	Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz
Architecture	x86_64
Microarchitecture	BDX
Family	6
Model	79
Stepping	1
Base Frequency	2.3GHz
Maximum Frequency	3.6GHz
All-core Maximum Frequency	2.9GHz
CPUs	72
On-line CPU List	0-71
Hyperthreading	Enabled
Cores per Socket	18
Sockets	2

Single-Core Turbo, All-Core Turbo, and Core Frequency

If svr-info is run with micro-benchmarks (by using -all) then the Health table in the HTML report will contain measured values for Single-Core Turbo and All-Core Turbo. The Core Frequency data will be shown in a chart.


System Health Inspector
File | C:/01_ServerInfo/svr-info/experiments/old/10.165.222.66.html#CPU
Intel® System Health Inspector
Configuration
Performance
Profile
Recommendations

Health

	10.165.222.66	Intel 2S Xeon E5-2699A v4 (reference)
CPU Speed	369439 ops/s	403415 ops/s
Single-core Turbo	3521 MHz	3509 MHz
All-core Turbo	2800 MHz	2980 MHz
Turbo TDP	259.52 Watts	289.90 Watts
Idle TDP	37.00 Watts	
Memory Peak Bandwidth	140.3 GB/s	138.1 GB/s
Memory Minimum Latency	79.4 ns	78 ns
Disk Speed	6633 iops	

Single-Core Turbo and All-Core Turbo are collected from a standard Linux utility called “turbostat”. Turbostat is executed in the background while a CPU intensive workload is running on one core or all-cores respectively.

There are two lines on the Core Frequency chart. One is the 'spec' frequency per active core which comes from same MSR as turbo values presented in the CPU table. The second line is the measured maximum frequency per active core. An internal utility is used to measure this.

Record of Updates

User Guide Revisions

Date	Revision	Description
06/30/2022	1.0	Initial draft
10/30/2022	2.1	svr-info_internal-2.0.1.tgz was changed to: svr-info-2.1.0.tgz In order to avoid confusion, the version of this user guide was changed to 2.1 to match the version of the software. Fully Supported Operating Systems updated on page 5: Ubuntu 16.04 and newer, CentOS 7 and newer. Note: svr-info may work on other Linux distributions, but has not been thoroughly tested Report formats were clarified on page 7. When Linux perf is installed on the target system, the new performance charts are available. See page 9. The JSON format has been updated on page 21 On page 25, clarified that ./reporter is in the /tools folder On page 28, clarified that ./collector is in the /tools folder
1/31/2023	2.2	Update for 2.2.0 release.
4/17/2023	2.3	Svr-info is now a single binary with tools and configuration files included inside the svr-info binary.

Feedback

We value your feedback. If you have comments (positive or negative) on this guide or are seeking something that is not part of this guide, please reach out and let us know what you think.

If you have information about a security issue or vulnerability with Intel® System Health Inspector, please send an e-mail to secure@intel.com. Encrypt sensitive information using our PGP public key.

Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.