

OPC UA: How many variables is the limit?

The title is a frequently asked question that can be answered quite easily: "There is no fixed limit."

Nevertheless, there are various possibilities to minimize the performance demand of the OPCUA server and thus to lighten the load on the controller. The influence of hardware is ignored in this article.

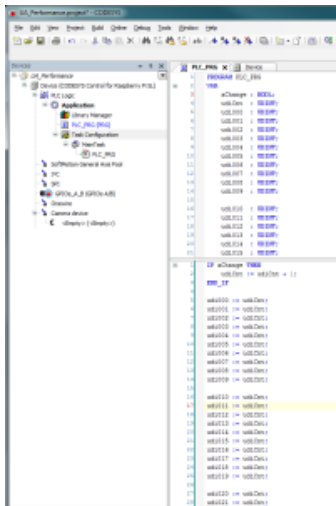


All values refer to the online mode. Load peaks during login or browsing of the data points via the client are not examined.

A Raspberry Pi 3 serves as controller and the Unified Automation UaExpert was used as client.

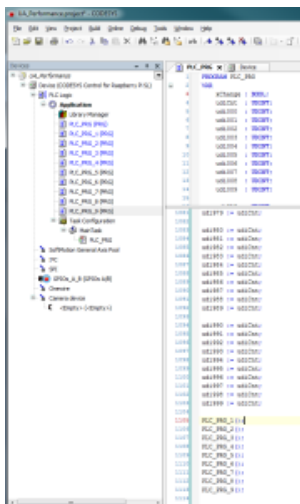
A POU with 1000 variables of type *UDINT* was created as a test project.

A value change can be switched on or off at the variables:



This POU can then be copied as often as required, working with 10,000 "prepared" variables.

The copied POUs are called using the code of *PLC_PRG*.



Simply providing the data points has no negative influence on the CPU load because the data is not used.
However, the compiler process takes longer.
In the test project, the CPU has a load of ~8%, whereby the monitoring of the CODESYS IDE is continuously active.

pi@ThK_raspberry: ~

top - 08:58:29 up 2:58, 1 user, load average: 1.12, 1.08, 1.05
Tasks: 112 total, 1 running, 111 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.6 us, 2.2 sy, 0.0 ni, 97.1 id, 0.0 wa, 0.0 hi, 0.1 si, 0.0 st
KiB Mem: 882500 total, 153200 used, 729300 free, 13528 buffers
KiB Swap: 0 total, 0 used, 0 free. 79600 cached Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
713	root	20	0	29876	23988	2228	S	7.9	2.7	19:57.22	codesyscontrol.
222	root	20	0	0	0	0	D	6.6	0.0	12:27.93	rcio_worker
197	root	20	0	0	0	0	S	1.3	0.0	2:30.92	spil
979	pi	20	0	5112	2540	2164	R	0.7	0.3	1:09.01	top

As soon as the client is connected, the load increases to ~9%, which is the base load.

Unified Automation UaExpert - The OPC Unified Architecture Client - PerformanceTest_1000*

File View Server Document Settings Help

Project Servers OPCUAServer@ThK_raspberry - None - None (uat) Documents Data Access View

Address Space No Highlight Root Objects

pi@ThK_raspberry: ~

top - 08:59:51 up 2:59, 1 user, load average: 1.09, 1.08, 1.05
Tasks: 112 total, 1 running, 111 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.8 us, 2.0 sy, 0.0 ni, 97.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 882500 total, 153200 used, 729300 free, 13560 buffers
KiB Swap: 0 total, 0 used, 0 free. 79600 cached Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
713	root	20	0	29876	23988	2228	S	9.6	2.7	20:03.95	codesyscontrol.
222	root	20	0	0	0	0	D	6.9	0.0	12:33.54	rcio_worker
197	root	20	0	0	0	0	S	1.3	0.0	2:32.07	spil
979	pi	20	0	5112	2540	2164	R	0.7	0.3	1:09.56	top

Value changes

When the first 1000 data points are activated by the client, the load increases to ~17%.

Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp
1	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt000	udt000	0	UInt32	11:04:19.769
2	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt001	udt001	0	UInt32	11:04:19.769
3	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt002	udt002	0	UInt32	11:04:19.769
4	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt003	udt003	0	UInt32	11:04:19.769
5	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt004	udt004	0	UInt32	11:04:19.769
6	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt005	udt005	0	UInt32	11:04:19.769
7	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt006	udt006	0	UInt32	11:04:19.769
8	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt007	udt007	0	UInt32	11:04:19.769
9	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt008	udt008	0	UInt32	11:04:19.769
10	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt009	udt009	0	UInt32	11:04:19.769
11	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt010	udt010	0	UInt32	11:04:19.769
12	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt011	udt011	0	UInt32	11:04:19.769
13	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt012	udt012	0	UInt32	11:04:19.769
14	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt013	udt013	0	UInt32	11:04:19.769
15	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt014	udt014	0	UInt32	11:04:19.769
16	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt015	udt015	0	UInt32	11:04:19.769
17	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt016	udt016	0	UInt32	11:04:19.769
18	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt017	udt017	0	UInt32	11:04:19.769
19	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt018	udt018	0	UInt32	11:04:19.769
20	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt019	udt019	0	UInt32	11:04:19.769
21	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt020	udt020	0	UInt32	11:04:19.769
22	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt021	udt021	0	UInt32	11:04:19.769
23	OPCUAServer@ThK_raspberry - None - None (uatcp-uasc-uabinary)	NS4[String]var CODESYS Control for Raspberry Pi SLApplication.PLC_PRG.udt022	udt022	0	UInt32	11:04:19.769

pi@ThK_raspberry: ~

top - 09:04:38 up 3:04, 1 user, load average: 1.24, 1.13, 1.07
Tasks: 112 total, 1 running, 111 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.7 us, 3.1 sy, 0.0 ni, 94.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 882500 total, 153480 used, 729020 free, 13588 buffers
KiB Swap: 0 total, 0 used, 0 free. 79600 cached Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
713	root	20	0	29876	23988	2228	S	16.5	2.7	20:38.03	codesyscontrol.
222	root	20	0	0	0	0	D	7.3	0.0	12:53.21	rcio_worker
197	root	20	0	0	0	0	S	1.3	0.0	2:36.09	spil
979	pi	20	0	5112	2540	2164	R	0.7	0.3	1:11.50	top

When we start the counter, we have a constant value change of the variable.
This increases the load to ~23%.

The screenshot shows a 'Data Access View' window with a table of OPCUA variables. The 'Value' column for several variables is highlighted with a red box, showing the value 1074. Below the table, a terminal window shows the output of the 'top' command, indicating a CPU load of 2.2%.

The measurement results with an increasing number of data points are summarized in the following table:

Number of data points	Subscribed data points without value change	Subscribed data points with value change
0	~10% (base load) of the project	~10% (base load) of the project
1000	~17%	~23%
2000	~25%	~35%
3000	~30%	~45%
4000	~36%	~55%
5000	~41%	~67%



The first conclusion can be drawn as follows:

- The CPU load, and therefore also the number of variables, depends on the number of value changes.

The next point, the sampling rate, can also be derived from this.

Sampling rate

The above measurement results are recorded with the default settings of the OPCUA client with an refresh rate of 500 milliseconds.
There are many values and parameters for which a slower refresh rate at the client has practically no influence.
As an example, a room temperature or preset/setpoints, such as the parameters of a PID controller, should be mentioned here.

Starting from the worst case in the above table, more and more data is now set to a lower sampling rate, while the others remain unchanged:

Sampling rate / Number of data points	0	1000	2000	3000	4000	5000
1000	~67%	~65%	~64%	~62%	~60%	~59%
2000	~67%	~64%	~62%	~59%	~58%	~56%
5000	~67%	~64%	~61%	~59%	~56%	~54%



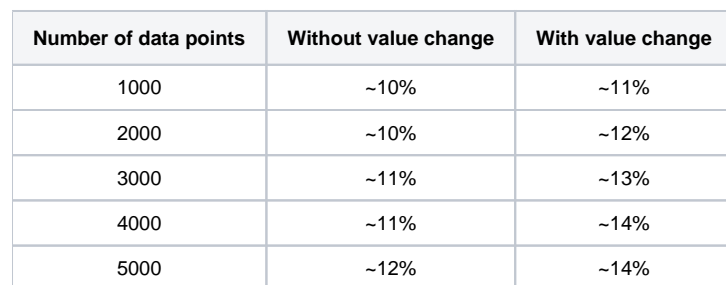
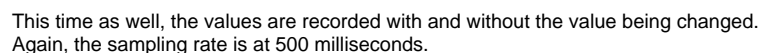
A mixed operation of 1000 variables each with a sampling rate of 500, 1000, 2000, 3000, and 4000 milliseconds resulted in a CPU load of ~59%.

The second conclusion can be drawn as follows:

- The CPU load can also be reduced by dividing the variables into groups with different refresh rates.

Combination of data points in one array

For demonstration purposes, the program is extended a little bit:



Even if combining data points in different groups/arrays means more work in the actual project, this investment should be made for larger plants.



Please note that all tests were performed with only one client connected.

Of course, every additional client also increases the CPU load.

Here the first measurement (1000 individual variables with a value change) with two connected OPCUA clients:

The screenshot displays two instances of the 'Unified Automation UaExpert - The OPC Unified Architecture Client - PerformanceTest_1000' application. The left window shows a project tree with 'Servers' and 'Documents' folders. The right window shows a 'Data Access View' with a table of data points. Below these windows, a terminal window shows the output of the 'top' command, indicating system performance metrics.

Terminal Output:

```
top - 12:37:18 up 6:37, 1 user, load average: 1.33, 1.29, 1.20
Tasks: 112 total, 1 running, 111 sleeping, 0 stopped, 0 zombie
%Cpu(s): 3.7 us, 1.9 sy, 0.0 ni, 94.1 id, 0.0 wa, 0.0 hi, 0.3 si, 0.0 st
KiB Mem: 882500 total, 156100 used, 726400 free, 15756 buffers
KiB Swap: 0 total, 0 used, 0 free, 79660 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
713	root	20	0	30232	24540	2228	S	23.8	2.8	106:31.13	codesyscontrol.
222	root	20	0	0	0	0	D	6.6	0.0	27:30.67	rcio_worker
197	root	20	0	0	0	0	S	1.7	0.0	5:31.53	spil
979	pi	20	0	5248	2540	2164	R	1.0	0.3	2:38.48	top