CYBERTEC

DATA SCIENCE & POSTGRESQL

POSTGRESQL

PROSPECTS ON LOW TCO ARM SYSTEMS

Kaarel Moppel
Senior Consultant
at CYBERTEC

I've been interested with databases for the last 9 years, working last 5 years exclusively with PostgreSQL. And still I'm constantly surprised by it's powerful set of features and the fast pace of development by the globally friendly community.

On a recent conference we came into contact with people working on an interesting hardware project called M2DC - the Modular Microserver Data Centre.

The project is about building modular, scalable, low-power servers (that could be clustered), that is developed in co-operation by a group of hardware and software companies - Toradex, Brytlyt, Christmann and the University of Bielefeld - and supported by European Union's Horizon 2020 research and innovation program.

End goal of the project: reduce TCO of exploiting servers by 50%.
So it seems to be quite a big development effort, but the first fruits are already available for general purpose applications (speed of thought: analytics, deep learning etc).

After asking nicely we got access to a cluster of 20 nodes based on Tegra TK1 SoC-s (32-bit 4 core ARM v7 CPU architecture), integrated all into one small 1U rack unit.

Why did the project catch our attention in the first place?
Because PostgreSQL is one of those few most widespread RDBMS-s that can run nicely on ARM architecture!

So let's see in more details what's inside the box and what can it do.

Tegra TK1 based cluster hardware

20 Apalis TK1 Compute Modules in 1U size
RECS|Box Compute Unit with following specs:

NVIDIA SoC	CD575M-A1
CPU Cores	4+1 (4 CPU + 1 Microcontroller)
ARM Cortex Version	A15
GPU	192 CUDA® Cores
L1 Instruction Cache (per core)	32KByte
L1 Data Cache (each core)	32KByte
L2 Cache (shared by cores)	2MByte
Maximum CPU frequency	2.2GHz
DDR3L RAM Size	2GByte
DDR3L RAM Speed	1866MT/s (64bit bus)
eMMC (SSD) disk	16GB
Network	1GbE
Max TDP	15 W

One such compute module looks something like that (picture right) and costs ~200 EUR (cheaper for larger quantities) for one module.

CYBERTEC
DATA SCIENCE & POSTGRESQL

www.cybertec-postgresql.com

Software

Nodes had Ubuntu 14.04.5 LTS installed on them and for our testing we used standard Postgres 9.6.1 compiled from sources.

Performance of compiling Postgres

Before getting to the real test, the first observation is already interesting: *How long does compiling on a single ARM node take compared to a similar amd64 server?*

Choosing a similar amd64 server setup here is very difficult though as comparing clock speeds doesn't tell the whole story (32 vs 64 bits, cache sizes) but scanning through AWS EC2 instance type specs seems that the most similar should be m3.xlarge. It has the same amount of cores (4) at slightly bigger clock speeds (2.5 GHz Xeon E5-2670 v2 vs 2.2 Ghz on Tegra, 14% difference) and also a lot more RAM (15GB vs 2GB), but for our case it shouldn't actually matter though as compiling Postgres only typically uses less than 300 MB of memory.

So here are the results:

	Make	Make -j 4
Tegra TK1	650s	205s
m3.xlarge	202s	91s

Well, what we can see is that compiling is actually a lot slower on Tegra. The result can't be explained by just the CPU clock speed difference.

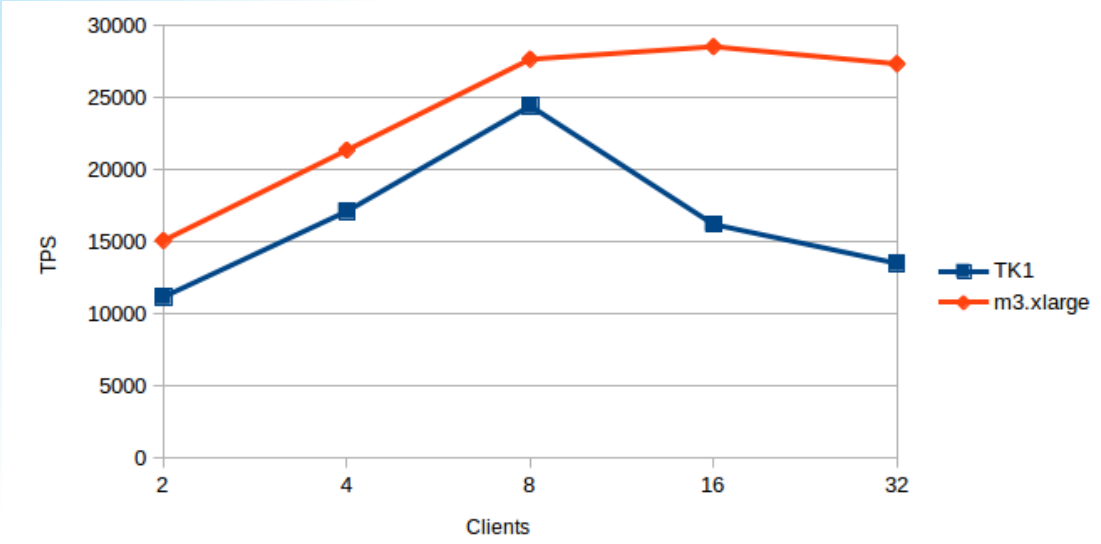
Might just be that compiling on RISC architecture is inherently slower due to fewer available CPU instruction (thus more workarounds) and more memory access. Also Intel server processors have quite big L3 caches which TK1 doesn't have.

Performance of pgbench read-only

Test 1 – read-only, scale 10, clients 2, 4, 8, 16, 32 (3x5min for each)

To remove element of disk access (nodes had small slowish SSD installed + NFS shares) and to see memory access/threading in action I decided to test with a small read-only dataset of scale=10, yielding 128 MB size for pgbench_accounts. Postgres shared buffers was configured at 512MB, all other settings left to default. Postgres version 9.6.1.

Here are the results:



What do we see?

Well, m3.xlarge still leads by average ~24% on low client numbers (2-8) but considering 14% higher clock speeds it's all good. But what happens at client counts 16 and 32?

A massive slowdown for the Tegra...how can one explain that?

… how … ?

Looking at the CPU load graph *(captured by the way with a new Postgres monitoring tool that we're hoping to release very soon)* seems that the CPU load is actually very light, staying below 8 at all times.

Hmm, what could be the reason then?

All the data is also nicely cached by Postgres (hit ratio 100% as seen from the graph) so we're not waiting for disk. After checking CPU load numbers also on the Intel machine (they were similarly low) my best bet is again pointing to architectural differences.

Meaning bigger penalties for heavier context switching scenarios (every Postgres client/backend runs as a separate process) and perhaps also on using synchronization primitives (locking) that Postgres is making use of to protect shared resources, like 16k of shared buffers in our case.

Conclusion

Drawing a conclusion is not easy here.

The Tegra-based node performed good for our read-only test for client counts to 8, being a tad slower but considering the fact that its power usage is also many times lower than of typical server nodes, you'll get way more performance per watt as with normal servers.

Direct number comparisons are again difficult but for example the Xeon E5-2670 v2 working in our m3.xlarge has a per CPU TDP of 115W (10 cores) but Tegra cluster has a maximum TDP of 15W per module (including 4 CPU cores + 192 GPUs + RAM), so it would be at least 4-5 times more efficient, project website itself stating 10-100x range.

So in the end I think such nodes used in a cluster setup could make a lot of sense for mass OLTP solutions with relatively simple operations as metrics ingestion from thousands of agents, IOT etc., where it comes down to cheap processing power.

For analytics though not too much, as memory is still somewhat limited and being 32-bits could turn into a bottleneck also.

So how could one make use of the cluster? For our test we just used a single node to test the basic capabilities, but a good way (in sense that no 3rd party or custom code needed) would be using Postgres Foreign Data Wrappers in combination with table inheritance for sharding.

Combined with some cheaper network storage one could really get a good price/performance ratio here. Best resource utilization for the nodes would be through making use of all the GPUs (192x CUDA cores) but setup of that is a bit complex (CUDA driver + PGStrom extension on Postgres side). Maybe we'll find some time soon to try out CUDA or FDW clustering and bring you another test.

Links with more info about the M2DC project

http://m2dc.eu/
https://www.toradex.com/computer-on-modules/apalis-arm-family/nvidia-tegra-k1
https://recswiki.christmann.info/wiki/doku.php?id=documentation:introduction#apalis_modules_arm

CONTACT

CYBERTEC PostgreSQL International GmbH

Gröhrmühlgasse 26
2700 Wiener Neustadt
Austria

Web: www.cybertec-postgresql.com

Phone: +43 (0)2622 93022-0
Fax: +43 (0)2622 93025
E-Mail: sales@cybertec.at

Visit us on:
Facebook | Twitter | LinkedIn | Xing | GitHub