MaxQuant goes Linux

To the Editor: We report a Linux version of MaxQuant[1] (http://www.biochem.mpg.de/5111795/maxquant), our popular software platform for the analysis of shotgun proteomics data.

One of our main intentions in developing MaxQuant was to 'take the pain out of' quantifying large collections of protein profiles[2]. However, unlike, for instance, the Trans-Proteomic Pipeline[3], the original version of MaxQuant could be run only on Microsoft Windows, and thus its use was restricted in high-performance computing environments, which very rarely use Windows as an operating system. When we began developing MaxQuant, Windows was the only operating system supported by vendor-provided raw data access libraries. Therefore, we wrote MaxQuant in the C# programming language on top of the Windows-only .NET framework. Windows support for cloud platforms is more expensive, and the operating system is harder to use and less scalable compared with Linux.

We recently carried out a major restructuring of the MaxQuant codebase, and we made it compatible with Mono (https://www.mono-project.com/), an alternative cross-platform implementation of the .NET framework. Furthermore, we now provide an entry point to MaxQuant from the command line without the need to start its graphical user interface, which allows execution from scripts or other processing tools. Meanwhile, Thermo Fisher Scientific has released its platform-independent and Mono-compatible implementation of its raw data access library (http://planetorbitrap.com/rawfilereader), and hopefully more vendors will follow soon. Together, this leads to a situation in which large-scale computing of proteomics data with MaxQuant becomes feasible on all common platforms.

When we parallelized the MaxQuant workflow over only a few central processing unit (CPU) cores, we hardly noticed a difference in performance between Linux and Windows (Fig. 1). However, in benchmarking of a highly parallelized MaxQuant run on 120 logical cores, we observed that the Linux version showed highly superior parallelization performance, with speed 64% faster than that observed under a Windows server operating system using identical hardware. MaxQuant uses operating system processes, rather than the intrinsic multi-threading mechanism of C#, to realize parallel execution, and it manages the load-balancing of an arbitrarily large set of raw data files over a specified number of processors by itself. We hypothesize that this allows Linux to optimize parallel execution to the high extent that we observed. A larger benchmark study is under way, in which we will investigate the dependence of the increased speed on hardware such as, for instance, the type of CPU and storage systems.

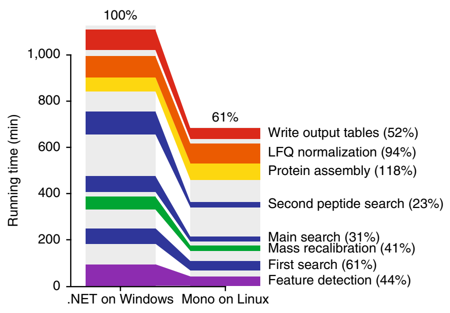

Fig. 1 | Benchmarking MaxQuant on Linux and Windows. We analyzed 300 LC-MS runs with MaxQuant using 120 logical cores in parallel, once with Ubuntu Linux (version 16.04.3) and once with Windows server 2012 R2 as the operating system. We used identical hardware in both cases: four Intel Xeon E7-4870 CPUs and 256 GB of DDR3 RAM. The total running times are shown, and several long-running sub-workflows are highlighted. Percentages indicate the amount of time needed to complete the relevant process in Linux as a percentage of the total time required for the same process in Windows.

MaxQuant has already been adapted in several forms for cloud and high-performance computing applications, as described, for instance, by Judson et al.[4] and on the Chorus platform (https://chorusproject.org). We expect that the number of applications will increase with our Linux-compatible MaxQuant version. We envision that proteomics core facilities, for instance, will benefit from the combination of command-line access and Linux compatibility, which enables standardized high-throughput data analysis. The MaxQuant code base is identical for Windows and for Linux; thus there is only a single distributable running on both operating systems, which can be downloaded from http://www.maxquant.org (version 1.6.1.0). MaxQuant is freeware, and contributions to new functionality are collaboration-based. The code of open source parts is available at https://github.com/JurgenCox/compbio-base. ❐

Pavel Sinitcyn, Shivani Tiwary, Jan Rudolph, Petra Gutenbrunner, Christoph Wichmann, Şule Yılmaz, Hamid Hamzeiy, Favio Salinas and Jürgen Cox*

Computational Systems Biochemistry, Max Planck Institute for Biochemistry, Martinsried, Germany.

**e-mail: cox@biochem.mpg.de*

Published online: 31 May 2018
https://doi.org/10.1038/s41592-018-0018-y

References

1. Cox, J. & Mann, M. *Nat. Biotechnol.* **26**, 1367–1372 (2008).
2. Azvolinsky, A., DeFrancesco, L., Waltz, E. & Webb, S. *Nat. Biotechnol.* **34**, 256–261 (2016).
3. Deutsch, E. W. et al. *Proteomics Clin. Appl.* **9**, 745–754 (2015).
4. Judson, B., McGrath, G., Peuchen, E. H., Champion, M. M. & Brenner, P. In *Proc. 8th Workshop on Scientific Cloud Computing* (eds. Chard, K. et al.) 17–24 (ACM, New York, 2017).

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation program (grant agreement no. 686547 to J.C., J.R. and S.Y.) and from the FP7 (grant GA ERC-2012-SyG_318987–ToPAG to S.T. and F.S.).

Author contributions

P.S., S.T., J.R., P.G., C.W., S.Y., H.H., F.S. and J.C. developed the software. P.S. conducted the performance analysis. J.C. wrote the manuscript.

Competing interests

The authors declare no competing interests.