clustermq enables efficient parallelization of genomic analyses

Schubert, Michael

Published in:
Bioinformatics

DOI:
10.1093/bioinformatics/btz284

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2019

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):
Schubert, M. (2019). clustermq enables efficient parallelization of genomic analyses. Bioinformatics, 35(21), 4493-4495. https://doi.org/10.1093/bioinformatics/btz284

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license. More information can be found on the University of Groningen website: https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment.

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Download date: 14-10-2023
Systems biology

**clustermq enables efficient parallelization of genomic analyses**

Michael Schubert

European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Trust Genome Campus, Cambridge, UK

*To whom correspondence should be addressed.
†Present address: European Research Institute for the Biology of Ageing, University of Groningen, University Medical Center Groningen, 9713 AV Groningen, the Netherlands

Abstract

Motivation: High performance computing (HPC) clusters play a pivotal role in large-scale bioinformatics analysis and modeling. For the statistical computing language R, packages exist to enable a user to submit their analyses as jobs on HPC schedulers. However, these packages do not scale well to high numbers of tasks, and their processing overhead quickly becomes a prohibitive bottleneck.

Results: Here we present clustermq, an R package that can process analyses up to three orders of magnitude faster than previously published alternatives. We show this for investigating genomic associations of drug sensitivity in cancer cell lines, but it can be applied to any kind of parallelizable workflow.

Availability and implementation: The package is available on CRAN and https://github.com/mschubert/clustermq. Code for performance testing is available at https://github.com/mschubert/clustermq-performance.

Contact: m.schubert@rug.nl

Supplementary information: Supplementary data are available at Bioinformatics online.

1 Introduction

The volume of data produced in the biological sciences has recently increased by orders of magnitude across many disciplines, most apparent in single cell sequencing (Svensson et al., 2018). In order to analyze this data, there is a need not only for efficient algorithms, but also for efficient and user-friendly utilization of high performance computing (HPC). Having reached a limit in the speed of single processors, the focus has shifted to distributing computing power to multiple processors or indeed multiple machines. HPC clusters have played and are continuing to play an integral role in bioinformatic data analysis and modelling. However, efficient parallelization using low-level systems such as MPI, or submitting jobs that later communicate via network sockets, requires specialist knowledge.

For the popular statistical computing language R (Ihaka and Gentleman, 1996) several packages have been developed that are able to automate parallel workflows on HPC without the need for low-level programming. The best-known packages for this are BatchJobs (Bischl et al., 2015) and batchtools (Lang et al., 2017). They provide a consistent interface for distributing tasks over multiple workers by automatically creating the files required for processing each individual computation, and collect the results back to the main session upon completion.

However, these packages write arguments and results of individual function calls to a networked file system. This is highly inefficient for a large number of calls and effectively limits these packages at about $10^6$ function evaluations (cf. Fig. 1 and Supplementary Methods). In addition, it hinders load balancing between computing nodes (as it requires a file-system based lock mechanism) and the use of remote compute facilities without shared storage systems.

Here we present the R package clustermq that overcomes these limitations and provides a minimal interface to submit jobs on a
Another way of parallelization is to register `clustermq` as a parallel `foreach` backend. This is particularly useful if a third-party package uses `foreach` loops internally, like all Bioconductor (Gentleman et al., 2004) packages that make use of BiocParallel.

    library(foreach)
    register_dopar_cmq(n_jobs=2, memory=1024)
    # this will be executed as jobs
    foreach(i = 1:3)
        BiocParallel::register(BiocParallel::DoparParam())
        BiocParallel::bplapply(1:3, sqrt)

In addition, the package provides a documented worker API that can be used to build tools that need fine-grained control over the calls sent out instead of the normal scatter-gather approach (cf. Supplementary Technical Documentation).

### 3 Evaluation

In order to evaluate the performance of `clustermq` compared to the `BatchJobs` and `batchtools` packages (cf. Supplementary Methods), we first tested the overhead cost for each of these tools by evaluating a function of negligible runtime and repeating this between 1000 and $10^6$ times. We found that `clustermq` has about $1000 \times$ less overhead cost compared to the other packages when processing $10^5$ or more calls, although across the whole range a clear speedup is apparent (Fig. 1a). The maximum number of calls that `BatchJobs` could successfully process was $10^5$, while `batchtools` increased this limit to $10^6$. By contrast, `clustermq` was able to process $10^7$ calls in about one hour.

For our second evaluation, we chose a realistic scenario with application to biological data. The Genomics of Drug Sensitivity in Cancer (GDSC) project published molecular data of approximately 1000 cell lines and their response (IC$_{50}$) to 265 drugs (Iorio et al., 2016). We ask the question if any one of 1073 genomic or epigenomic events (mutation/copy number aberration of a gene and differential promoter methylation, respectively) is correlated with a significant difference in drug sensitivity across all cell lines or for 25 specific cancer types ($n = 7,392,970$ associations). We found that for this setup, `clustermq` is able to process the associations in about one hour with 10% lost to overhead (Fig. 1b; dashed line). The other packages produced too many small temporary files for our networked file system to handle, and by extrapolation processing all associations would have taken over a week.

To achieve similar results using the previously published packages one would need to adapt the analysis code to chunk together related associations and explicitly loop through different subsets of data. `clustermq` lifts this requirement and lets the analyst focus on the biological question they are trying to address instead of manually optimizing code parallelization for execution time (cf. Supplementary Discussion).

### 4 Conclusion

The `clustermq` R package enables computational analysts to efficiently distribute a large number of function calls via HPC schedulers, while reducing the need to adapt code between different systems. We have shown its utility for drug screening data, but it
can be used a broad range of analyses. This includes Bioconductor packages that make use of BiocParallel.

Conflict of Interest: none declared.

References

Bischl,B. et al. (2015) BatchJobs and BatchExperiments: abstraction mechanisms for using R in batch environments. *J. Stat. Softw.*, 64, 1–25.

Gentleman,R.C. et al. (2004) Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.*, 5, R80.

Ihaka,R. and Gentleman,R. (1996) R: a language for data analysis and graphics. *J. Comput. Graph. Stat.*, 5, 299–314.

Iorio,F. et al. (2016) A landscape of pharmacogenomic interactions in cancer. *Cell*, 166, 740–754.

Lang,M. et al. (2017) batchtools: tools for R to work on batch systems. *J. Open Source Softw.*, 2, 135.

Svensson,V. et al. (2018) Exponential scaling of single-cell RNA-seq in the past decade. *Nat. Protoc.*, 13, 599–604.

Hintjens,P. (2013) ZeroMQ: Messaging for Many Applications. O’Reilly Media, Inc, Sebastopol, California.

Svensson,V. et al. (2018) Exponential scaling of single-cell RNA-seq in the past decade. *Nat. Protoc.*, 13, 599–604.