A Comparative Study of Parallel Programming Language Support for Bioinformatics

Project under supervision of Dave Clarke
dave.clarke@it.uu.se

May 28, 2014

Abstract

The goal of this project is to evaluate the applicability of a number of parallel programming languages/frameworks for developing bioinformatics algorithms, considering issues such as easy-of-programming, efficiency, scalability, and composability.

Bioinformatics is the study of biological processes using computers. At the core of bioinformatics are algorithms that operate on a representation of DNA, namely strings such as the following:

```
ACAATGAGGTCACTATGTTCGAGCTCTTCAAACCGGCTGCGCATACGCAGCGGCTG
CGCCTTCGTTGGCAACTTTTCATCGGTATTTTTGTTGGCTATGCAGGCTACTATTT
TGATGGGGAACGTCTCTGACCGTTCTAACCCGCGCTACTTTCTGAGTGCAGGTCTA
GGCTTTATGCCATGGGCAACGGGCAGCATTACTGCGATGTTTATTCTGCTGTTCTT
...
```

Such data is typically large (many gigabytes) and flawed. Algorithms operating on such strings look for recurring patterns or commonality between different strings, perform statistical calculations, and so forth, to extract information from (sets of) such strings in order to answer hypotheses posed by biologists.

Getting the underlying algorithms fast is crucial for many bioinformatic tasks. To this end, multicore (and manycore) computers offer a way forwards, as vast amounts of parallel processing power are now readily available. Exploiting parallel computing resources requires parallel programming languages, and many such languages have been devised.

One of the key challenges is that the bioinformatics programs are typically written by bioinformaticians (biologists), not computer programmers. Consequently, programmability is often considered more important than efficiency. Typically, biologist will write a program in an easy-to-use programming language (such as Python or R) building upon existing libraries. Programming time will be short, but the resulting program may run for weeks to deliver a result. A computer scientist approaching the same problem would program for several weeks to produce a program that runs in a short amount of time. The

language used by the computer scientist would typically require deep knowledge of the hardware and compiler in order to achieve efficiency. Thus there is a trade-off between programmability and efficiency. Research, for example, in the UpScale project, is working towards bridging that gap. The role of the present project is to evaluate the features of existing programming languages in terms of how well they support bioinformatic algorithms.

A second key issue is scalability. Will algorithms/languages/compilers/runtimes/machine infrastructure support increasing data sizes while maintaining a linear increase in run-time?

A third key issue is composability. Bioinformaticians typically build workflows composing several different algorithms to solve their problems. Often these algorithms have different requirements on their input data, and sometimes they are only available in different languages, which means that vast amounts of data needs to be transformed or even saved to file before it can be used in the next step of a workflow. The core question is what support do languages provide for composability and how does this impact performance?

The project will take the following steps:

Core algorithms Initially a number of core bioinformatic algorithms will be implemented. This will also familiarise student with bioinformatics. (See `http://rosalind.info/` for many examples.)

Parallelisation Next, a selection of these algorithms will be made as targets for parallelisation and further investigation. The algorithms chosen should represent the diversity of bioinformatic algorithms. These will be implemented in languages/frameworks such as Erlang, Haskell (+ parallel programming extensions), Java+Fork-Join, Java+ABS-API, and Open MP.[1]

Composition True bioinformatic algorithms consist of workflows composing several algorithms. This step will explore how the languages support for such workflows. Issues considered will include the demands on programmer to perform costly data transformations and the language support for this.

Evaluation & benchmarking Ultimately, the project is about evaluating the suitability of various language/framework features for implementing bioinformatics algorithms and workflows. Factors to consider when doing the evaluation and will include readability and conciseness of the code, parallelisability of the problem, adaptability of the algorithm, for instance, to different machine configurations, support for composition, and scalability.

The student taking on this project should have interest in programming languages, algorithms and parallel computing, and skill in programming in multiple programming languages. Knowledge of biology or bioinformatics would be helpful, but is certainly not required.

[1] The precise choice of languages is open for discussion.