Parallelization and Grid Computing

Thilo Kielmann
kielmann@cs.vu.nl

Bioinformatics Data Analysis and Tools
June 12th, 2007
Jaap Heringa said:

- There are two extremes in bioinformatics work
  - **Tool users (biologists):** know how to press the buttons and the biology but have no clue what happens inside the program
  - **Tool shapers (informaticians):** know the algorithms and how the tool works but have no clue about the biology

*Both extremes are dangerous, we need a breed that can do both*
Thilo Kielmann says:

I am even more extreme:

• I know nothing (scientific) about biology

• I am not even one of Jaap's *informatitions* who know the algorithms and how the tools work

• I am a computer scientist who knows how computers work and can (could) be used by bioinformatics
(Bio-)Informatics

• **Informatics**: includes the science of information and the practice of information processing

• Used as a compound, as in **bio-informatics**, it denotes the specialization of informatics to the management and processing of data, information and knowledge in the named discipline.
Computational Science

• ...is concerned with constructing mathematical models and numerical solution techniques and using **computers to** analyze and **solve scientific** and engineering **problems**.

• In practical use, it is typically the application of **computer simulation** and other forms of computation to problems in various scientific disciplines.
Scientists develop application software that model systems being studied and run these programs with various sets of input parameters.

Typically, these models require massive amounts of calculations and are often executed on supercomputers or distributed computing platforms.

Example: SnapDRAGON
Supercomputers

If you have to ask for the price...
(see also: www.top500.org)
Computer Clusters

• A computer cluster is a group of loosely coupled computers that work together closely so that in many respects they can be viewed as though they are a single computer.

• Clusters are commonly, but not always, connected through fast local area networks.

• Clusters are usually deployed to improve speed and/or reliability over that provided by a single computer, while typically being much more cost-effective than single computers of comparable speed or reliability.
The DAS-2 Cluster at the VU
Parallel Computing

• It's all about speed!

• If your own PC gets your work done, you don't need to think about using more than one computer...

• However, if your PC is too slow:
  – Profile your program; find out \( \textit{which part} \) takes most of the time

  2. Improve your algorithms; getting from \( O(n^3) \) down to \( O(n^2) \) helps more than adding another PC
A Million-Fold Speed Improvement in Genomic Repeats Detection

John W. Romein, Jaap Heringa, and Henri E. Bal

Vrije Universiteit
Faculty of Sciences, Department of Computer Science
Bio-Informatics Group & Computer Systems Group
Amsterdam, The Netherlands

{john, heringa, bal}@cs.vu.nl

Abstract

This paper presents a novel, parallel algorithm for generating top alignments. Top alignments are used for finding internal repeats in biological sequences like proteins and genes. Our algorithm replaces an older, sequential algorithm (Repn), which was prohibitively slow for sequence lengths higher than 2000. The new algorithm is an order of magnitude faster (O(n^3)) rather than O(n^4).

The paper presents a three-level parallel implementation of the algorithm: using SIMD multimedia extensions found on present-day processors (a novel technique that can be used to parallelize any application that performs many sequence alignments), using shared-memory parallelism, and using distributed-memory parallelism. It allows processing the longest known proteins (nearly 35,000 amino acids). We show exceptionally high speed improvements: between 548 and 889 on a cluster of 64 dual-processor machines, compared to the new sequential algorithm. Especially for long sequences, extreme speed improvements over the old algorithm are obtained.

Gene duplication can take place at the level of copying complete genomes consisting of billions of nucleotides, down to only two or three nucleotides. Roughly, more than half the number of base pairs in most genomes are part of a repeat [4], which shows the importance of these copying mechanisms as a general mechanism for evolution. However, pathologically repeated fragments are also known to play a role in serious diseases like Huntington's.

Recognizing repeats in protein sequences is important, since they can reveal much information about the structure and function of a protein. Often, divergent evolution in repeats has blurred the ancestral ties, so that at first sight, the repeats hardly show any resemblance anymore. Frequently, only 10–25% of the amino acids in a repeated protein subsequence are conserved. Also, the repeats may have different lengths through insertions and deletions. Moreover, two repeats need not be consecutive (tandem), but may be interspersed by other subsequences (that can contain different repeats as well). These proper-
Data Processing
Schematic Program Code

Program:

```c
for ( i = 0; i < N; i++){
    input = in_file[i];
    result = compute(input);
    write(outfile,result);
}
```

what to do in parallel?
compute() or the loop?
Multi-phase Data Processing
Workflow / Taskflow

Combine programs to a complex parameter study.
Parallelization

Algorithm
- Write a new, parallel algorithm
- Communication is message exchange between parallel processes
- Fine-grained

Workflow
- Write a “wrapper” around existing, sequential algorithms
- Communication is program startup and file I/O
- coarse-grained
Workflow Example: Taverna

- http://taverna.sourceforge.net/
- Collaboration between European Bioinformatics Institute (EBI) and others
Using Workflow Systems

• You can focus on the biology
• Many (system-level) details are hidden from you
• But:
  – The workflow program (e.g., Taverna) must be able to run your tasks on your platform (e.g. DAS-3)
  – Workflow systems have their own learning curve
Commercial Break:

• Anybody up for doing a (M.Sc.) project?
• Using Taverna on the new DAS-3 (and/or on the bioinformatics cluster)
  – Make Taverna submit jobs to one or more DAS-3 clusters (see also remainder of talk)
  – Taverna is open source, written in Java
  – Investigate the usefulness of Taverna for certain bioinformatics problems
  – Ideal for a students who know a little of both worlds...
Using a Cluster

- Cluster consists of
  - Compute nodes
  - Frontend machine

- Frontend:
  - File server
  - Compile server
  - **scheduler**
A Scheduler?

• Assigns compute nodes to programs
  – Each node runs (ideally) one program only “space sharing”
  – Users request number of compute nodes for a certain time interval (“16 nodes for 30 minutes”)
  – Programs (“jobs”) wait in queues queues may have different priorities

• Scheduler is planning and executing jobs from the queues
Important Scheduler Commands

• qsub
  submit a new job

• qdel
  delete a queued job

• qstat
  ask status about current jobs

• De-facto standard (used by PBS, SGE,...)
Writing your own Workflow

• Use your favourite language/tool!
• (Shell) script running on the front end, mostly calling qsub
• C/C++/Java/Python program, running on the front end, mostly calling qsub
• C/C++/Java/Python program, running on a compute node, calling programs on other compute nodes
  – Needs a qsub in advance for the total number of nodes to be used
Writing your own Parallel Algorithms

• Needs more solid background in Computer Science
  – Course “Parallel Programming” by Henri Bal, given in the fall terms

• Improves the speed of a single program, mostly in a fine-grained way

• Might not be necessary as long as parallelism can be achieved calling the sequential algorithm in parallel, with different data
Master-worker Parallelism

- Hybrid form between
  - Workflow
  - Parallel algorithm
- One *master* program is assigning *tasks* to *worker* programs
- Tasks are independent, trivial parallelism
  - Typical: tasks operate on different data, like comparing different genome sequences with a genome database
Grid Computing
What is the Grid?

• According to Ian Foster's *Three Point checklist*:

• A grid is a system that:
  1. Coordinates resources that are not subject to centralized control...
  2. ....using standard, open, general-purpose protocols and interfaces...
  3. ....to deliver non-trivial qualities of service.
Grids and Virtual Organizations (VO's)

• There are many grids:
  – Resources contributed by their owners and integrated for sharing:
    • Compute clusters
    • Data repositories
    • Instruments (radio telescopes, particle accelerators)

• Virtual Organizations
  – Communities sharing their resources
    • Provide users with certificates ("id's for login")
Example: the DAS-3 (just got operative)
Applications in Grids

Diagram showing the interactions between a user portal, resource brokering service, monitoring service, information service, and applications, with file I/O and communication pathways.
Functional Aspects of Grid Applications

• What applications need to do:
  – Access to compute resources, job spawning and scheduling
  – Access to file and data resources
  – Communication between parallel and distributed processes
  – Application monitoring and steering
Non-functional Aspects of Grid Applications

• What else needs to be taken care of:
  − Performance
  − Fault tolerance
  − Security and trust
  − Platform independence

• Grids are like the “wild west:”
  − Don't trust anything to work reliably.
  − Things keep changing all the time.
  − The application that worked yesterday, may very well fail today.
The Grid Application Toolkit (GAT)

Shield application from complex/heterogeneous/hostile Grid environment
GAT API Scope

• What's in it for you:
  – Files
  – Resources (compute nodes)
  – Event/Information Exchange
  – Utility classes
    (error handling, security, preferences...)
  – **Nothing else.**
    • (keep it simple)
GAT: a Simple Example

• Reading a remote file
  – Specified by URI (e.g., any://fs0.das2.cs.vu.nl/myfile)

```cpp
try
{
    char data[25];

    GAT::File file (context, source_url);

    file.open  (RD_ONLY);
    file.seek  (100, SEEK_SET);
    file.read  (data, sizeof(data));
    file.close ();
}

catch (GAT::Exception const &e)
{
    std::cerr << "Some error:" << e.what() << std::endl;
    return e.Result();
}
```
GAT Implementations

• See www.gridlab.org/gat
• Engine implementations:
  - C, wrappers for C++, Python
  - Java (done at VU)
• Adaptors for Globus, ssh, etc.
Summary

• If your computer is too slow, do it in parallel:
  – Workflows of sequential algorithms
  – Parallel algorithms (e.g., master/worker)
  – Both

• Clusters are widely available.

• Grids integrate clusters and data repositories.

• Writing grid applications is non-trivial, but might be worth the effort.