A Pipeline Processing Approach to GIS

M. Koutroumpas    C. Higgins

Edina
The University of Edinburgh

AGILE, 2008
Outline

1. Introduction
2. Implementation
3. Results
4. Conclusion
Introduction

- **Name:** Michael Koutroumpas
- **Organisation:** EDINA
A National Data Centre for Tertiary Education since 1995, based at the University of Edinburgh

Our mission... to enhance the productivity of research, learning and teaching in UK higher and further education

- Focus is on services but also undertake R&D
- Turn projects ⇒ services
- Substantial experience in handling geospatial data
SEcurE access to GEOspatial services

- Aiming to demonstrate how access to GI on Grid may be achieved:
  - Shibboleth
  - WS-Security
  - GSI
  - OGC Web Services

- Aims at promoting interoperability and standards

- This work is an offshoot of the R&D involved in the SEE-GEO project.
Each colour represents a different Geographical feature.
By splitting a slow process into many smaller ones, we can accelerate processing.
A processing pipeline consists of a serial chain of data processing tasks
A typical operation has thousands of features. We do feature level parallelization.
Time to output each feature is $\max_{1 \leq i \leq n} t_i$. 
Each colour represents a different Geographical feature.
By splitting a slow process into many smaller ones, we can accelerate processing.
A processing pipeline consists of a serial chain of data processing tasks
A typical operation has thousands of features. We do feature level parallelization.
Time to output each feature is $\max t_i$. $1 \leq i \leq n$
Each colour represents a different Geographical feature.

By splitting a slow process into many smaller ones, we can accelerate processing.

A processing pipeline consists of a serial chain of data processing tasks.

A typical operation has thousands of features. We do feature level parallelization.

Time to output each feature is $\max_{1 \leq i \leq n} t_i$. 

Each colour represents a different Geographical feature.
By splitting a slow process into many smaller ones, we can accelerate processing.
A processing pipeline consists of a serial chain of data processing tasks.
A typical operation has thousands of features. We do feature level parallelization.
Time to output each feature is $\max_{1 \leq i \leq n} t_i$. 
Each colour represents a different Geographical feature.
By splitting a slow process into many smaller ones, we can accelerate processing.
A processing pipeline consists of a serial chain of data processing tasks
A typical operation has thousands of features. We do feature level parallelization.
Time to output each feature is $\max_{1 \leq i \leq n} t_i$. 
Each colour represents a different Geographical feature.
By splitting a slow process into many smaller ones, we can accelerate processing.
A **processing pipeline** consists of a serial chain of data processing tasks
A typical operation has thousands of features. We do feature level parallelization.
Time to output each feature is $\max_{1 \leq i \leq n} t_i$. 
Each colour represents a different Geographical feature.
By splitting a slow process into many smaller ones, we can accelerate processing.
A processing pipeline consists of a serial chain of data processing tasks
A typical operation has thousands of features. We do feature level parallelization.
Time to output each feature is $\max_{1 \leq i \leq n} t_i$. 
Each colour represents a different Geographical feature.
By splitting a slow process into many smaller ones, we can accelerate processing.
A processing pipeline consists of a serial chain of data processing tasks
A typical operation has thousands of features. We do feature level parallelization.
Time to output each feature is $\max_{1 \leq i \leq n} t_i$. 

Pipeline Processing

- Each colour represents a different Geographical feature.
- By splitting a slow process into many smaller ones, we can accelerate processing.
- A processing pipeline consists of a serial chain of data processing tasks.
- A typical operation has thousands of features. We do feature level parallelization.
- Time to output each feature is $\max_{1 \leq i \leq n} t_i$. 

$\text{Forth}$
Each colour represents a different Geographical feature.

By splitting a slow process into many smaller ones, we can accelerate processing.

A processing pipeline consists of a serial chain of data processing tasks.

A typical operation has thousands of features. We do feature level parallelization.

Time to output each feature is \( \max_{1 \leq i \leq n} t_i \).
Pipeline Processing

- Each colour represents a different Geographical feature.
- By splitting a slow process into many smaller ones, we can accelerate processing.
- A **processing pipeline** consists of a serial chain of data processing tasks
- A typical operation has thousands of features. We do feature level parallelization.
- Time to output each feature is $\max_{1 \leq i \leq n} t_i$. 

![Diagram of pipeline processing with three circles labeled 10s each connected by arrows, indicating sequential processing stages.]

---

**EDiNA**
Pipeline Processing

- Each colour represents a different Geographical feature.
- By splitting a slow process into many smaller ones, we can accelerate processing.
- A processing pipeline consists of a serial chain of data processing tasks.
- A typical operation has thousands of features. We do feature level parallelization.
- Time to output each feature is $\max_{1 \leq i \leq n} t_i$. 
Each colour represents a different Geographical feature.
By splitting a slow process into many smaller ones, we can accelerate processing.
A processing pipeline consists of a serial chain of data processing tasks.
A typical operation has thousands of features. We do feature level parallelization.
Time to output each feature is $\max_{1 \leq i \leq n} t_i$. 
Is that a Workflow?

- A workflow is just a method for modelling a sequence of processes.
- Became well-known for modelling business processes with workflow engines.
- A processing pipeline is a method of parallelization.
- Became well-known with RISC processors.
- Workflows are not about parallelization.
The big picture...

Existing Tools

- **Workflows** are used to model the data flows between processes.
- Workflows can be used for dependency analysis, since they describe the flow of the data.
- There is extensive research for parallelizing processes.
- As well as standard ways of measuring performance like **throughput** and **speedup**.
- (Recently) The advent of middleware that support multithreading.
  - Used extensively in the Grid world and in Bioinformatics.
  - No existing applications in GIS with the aim of parallelizing them.
The big picture...

Missing

- A strict theoretical model that defines the prerequisites for a GIS process to be executed in a Workflow
- Practical steps to implement it in existing workflow middleware.
- A working implementation.

GOTCHA: You need to have an atomic unit to utilise a pipeline and overcome the flow dependencies.
A strict theoretical model that defines the prerequisites for a GIS process to be executed in a Workflow

Practical steps to implement it in existing workflow middleware.

A working implementation.

GOTCHA: You need to have an atomic unit to utilise a pipeline and overcome the flow dependencies.
Our Outputs

- A set of rules to follow and programming guidelines to make Parallelizable workflows for GIS operations
- Optimisation guidelines and metrics.
- A usable Implementation. Under a BSD-like licence.
- Demonstrators that use it.
A Concrete Example

- Participated in the OGC Interoperability experiment, aiming at the finalisation of
  - The Geolinked Data Access Service (GDAS) Draft
  - The GeoLinking Service (GLS) Draft
- A Geolinking service that joins geographical attributes with census data.
- It was a great opportunity to put the model into test.
A Concrete Example

- Census data can be from any GDAS compliant provider
- Feature data can be any WFS compliant server
- USE CASE: A **Geolinking** service that joins boundary geographies with census data.
- Note the number of tasks in the pipeline and the times.
Workflow engine

- Any workflow engine that is multithreaded can be used.
- We used OGSA-DAI as a workflow engine
- Well known grid middleware part of the Globus toolkit
Activities

- We need to extend it in a way that allows parallel processing without conflicts.
- In OGSA-DAI workflows can are constructed as a set of interconnected activities.

Requirements

- We need to take advantage of the atomic properties of features.
- We need to use buffered streams to communicate among the processes. No extra resources should be shared between them.
Results

Two thematic Maps of Leeds Combining Geographical Data

Boundary Data from UKBorders – provides boundary datasets of the UK

CasWeb at MIMAS – a Census Data provider.
Each activity runs on a different CPU. Therefore...
...it cannot scale more than the number of activities.
Specific details are in
http://edina.ac.uk/projects/seesaw/seegeo/about_benchmarking.html
Each activity runs on a different CPU. Therefore...
...it cannot scale more than the number of activities.

Specific details are in
http://edina.ac.uk/projects/seesaw/seegeo/about_benchmarking.html
Each activity runs on a different CPU. Therefore...
...it cannot scale more than the number of activities.
Specific details are in
http://edina.ac.uk/projects/seesaw/seegeo/about_benchmarking.html
Conclusion

By using such geo-enabled workflow engines we

- ...get a framework that is extensible and boosts reusability
- ...utilise multiple processors and parallelize our application for free.
- ... and we have a theoretical model to formally analyse, optimise and measure improvements in speed of GIS operations.

Both OGSA-DAI and our extensions are open source

Available at

http://edina.ac.uk/projects/seesaw/seegeo
QUESTIONS ???

Contact Details
- Name: Michael Koutroumpas
- Email: m.koutroumpas_AT_ed.ac.uk
- Website: http://edina.ac.uk/projects/seesaw/seegeo