SymGrid-Par

Coordinating Computer Algebra Systems for Parallel Symbolic Computation

Jost Berthold

University of St.Andrews
St.Andrews, Fife, UK

SCIEnce Workshop, Paris, Jan 2009

Outline

1. Introduction
2. High-Level Parallel Programming
   - Why we use functional programming
   - Algorithmic Skeletons
3. Implementation work
   - System Architecture
   - Current Status
4. Outlook and Summary

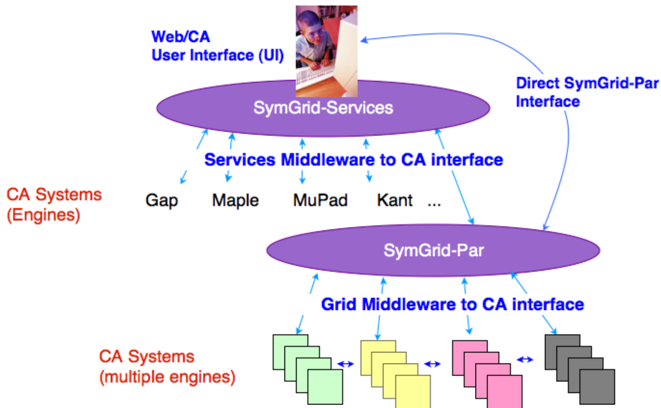
★ SCIENCE Symbolic Computation Infrastructure for Europe ★

Outline

1. Introduction
2. High-Level Parallel Programming
   - Why we use functional programming
   - Algorithmic Skeletons
3. Implementation work
   - System Architecture
   - Current Status
4. Outlook and Summary

SCIENCE
Symbolic Computation Infrastructure for Europe

Grid-Enabled Parallel Symbolic Computation



SCIENCE Symbolic Computation Infrastructure for Europe

SymGrid-Par

- Parallelising computer algebra (CA) systems...
- ...for better performance and distributed resource usage.
- Using a functional language, **high-level parallelism**,
- to provide a middleware for coordinated parallel CA execution.

Results to date:

- Design and prototype implementation interfacing parallel Haskell and several CA systems.
- Performance competitive to existing parallel CA solutions like parGAP. Superior for irregular problems.

Next steps:

- Using SCSCP, moving towards non-expert use.

SCIENCE
Symbolic Computation Infrastructure for Europe

Outline

1 Introduction

2 High-Level Parallel Programming
- Why we use functional programming
- Algorithmic Skeletons

3 Implementation work
- System Architecture
- Current Status

4 Outlook and Summary

SCIENCE
Symbolic Computation Infrastructure for Europe

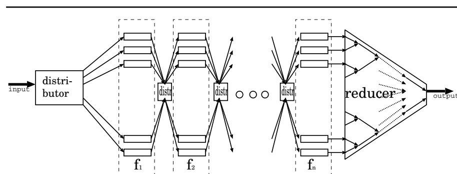
Why functional?

- Problems of mathematical nature.
- Higher-order functions.
- Sophisticated type systems.
- Modularity, code reuse.
- Computation structure exposes inherent parallelism.
- Implicit control flow and synchronisation.

Transitive closure under F:

$$\text{orbit}(S, F) = R \;\Leftrightarrow\; \forall_{r \in R}.\forall_{f \in F}.f(r) \in R$$

```
orbit s fs = iterateUntil isTheSame allF s
  where allF xs = union xs
                          (nub [ f x | f<-fs, x<-xs])
```

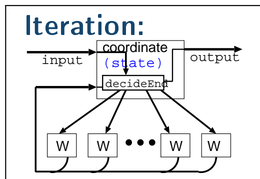


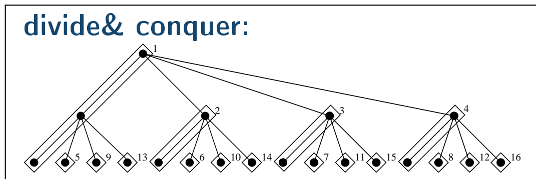
15 years of expertise in parallel concepts
Parallel + functional = High-level parallel programming

SCIENCE Symbolic Computation Infrastructure for Europe

Algorithmic Skeletons for Parallel Programming

- abstract specification of algorithm structure
- **focus on algorithm structure** as **higher-order function**
- abstract over concrete tasks (embedded "worker" functions),
- hidden parallel optimised implementation(s) (machine-specific)





Boxes and lines – **executable!**

Enable a high-level view on parallel systems and computations

SCIENCE
Symbolic
Computation
Infrastructure for
Europe

Common skeletons

- **Parallel transformation**: Map

  ```
  map :: (a -> b) -> [a] -> [b]
  ```

  independent **elementwise transformation**

- **Parallel Reduction**: Fold

  ```
  fold :: (a -> a -> a) -> [a] -> a
  ```

  with **commutative** and **associative** operation.

- **Parallel Map-Reduce**:

  ```
  parmapReduce :: (in -> [(k,temp)])
                  -> (k -> [temp] -> out)
                  -> [in] -> [(k,out)]
  ```

  transformation and **groupwise reduction**.

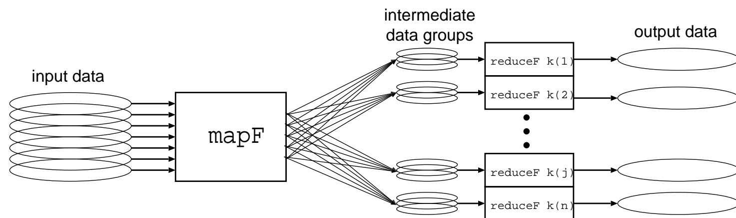
- **Parallel Scan**:

  ```
  parScanL :: (a -> a -> a) -> [a] -> [a]
  ```

  reduction keeping the intermediate results.

SCIENCE
Symbolic Computation Infrastructure for Europe

Parallel Map-and-Reduce

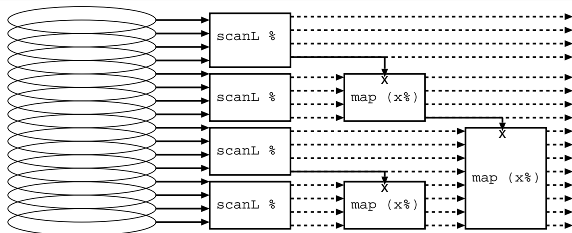



```
parmapReduce :: (in -> [(k,temp)])
             -> (k -> [temp] -> out
             -> [in] -> [(k,out)]
```

- made fashionable by Google employees ("Google Map-Reduce")
- combines map and fold, key-based grouping in-between
- can express various AI problems

SCIENCE Symbolic Computation Infrastructure for Europe

Parallel (left) Scan




```
parScanL :: (a -> a -> a) -> [a] -> [a]
```

- tree-based sub-reduction, keeping intermediate results
- sometimes a better alternative to `fold`
- Please note: **needs input length**

```
parScan lcm [3,6,2,15,23,138] = [3,6,6,30,690,???]
   where lcm a b = a * b `div` gcd a b
         gcd a b = if b == 0 then a else gcd b (a `rem` b)
```

SCIENCE Symbolic Computation Infrastructure for Europe

More algorithm-oriented and domain-specific

- **Divide and conquer**

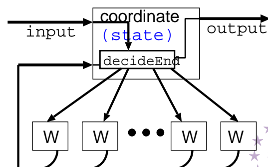
```
divCon :: (a -> Bool) -> (a -> b)      -- trivial? / then solve
          -> (a -> [a]) -> ([b] -> b) -- split / combine
          -> a -> b                   -- input / result
```

- **Iteration**

```
iterateUntil :: (in -> Int -> ([ws],[t],ms)) ->     -- split/init function
                (ws -> t -> (r,ws)) ->              -- worker function
                (ms -> [r] -> Either out ([t],ms)) -- manager function
                -> in -> out
```

Example: Orbit calculation

```
orbit :: [a] -> [a -> a] -> [a]
orbit   start  generators = iterateUntil ...
```



Implementation aspects: How many elements, how many iterations to expect? Are there enough generators to parallelise over them? How large will the objects become?

Outline

1. Introduction
2. High-Level Parallel Programming
   - Why we use functional programming
   - Algorithmic Skeletons
3. Implementation work
   - System Architecture
   - Current Status
4. Outlook and Summary

SCIENCE Symbolic Computation Infrastructure for Europe

Goal and Requirements

Earlier prototype:

- Very good performance, superlinear speedups,
- deals well with irregular problems,

  but...

- is not using SCSCP, but proprietary solutions.

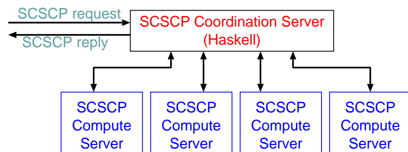
Use SCSCP-enabled compute servers!
Use high-level parallelism in Haskell!

- Haskell SCSCP API,
- Haskell Coordination Server,
- Custom domain-specific skeleton implementations,
- SCSCP symbols, content dictionary for Skeletons
- Advanced load balancing, maximum performance, GRID, ...

System Architecture

Coordination Server

- providing SCSCP skeleton symbols
- acting as server for users
- accessing Compute Servers (CAS) as a client



- Written in Haskell (type-safe functional language, automatic memory management, lazy evaluation)
- Skeletons directly realised as higher-order functions
- Coordinated parallel CAS execution with implicit synchronisation and control flow.

SCIENCE Symbolic Computation Infrastructure for Europe

Current Status

Synergies by joint research:

- Improving and stabilising the SCSCP specification
- Testing and stream-lining other SCSCP compute servers
- Identifying useful domain-specific skeletons.

Stepping stones:

- Prototype version (parallel Haskell system)
- SCSCP binding
- Single-server (synchronous) client API
- Asynchronous, and multi-server client API
- Coordination server infrastructure

  ——— (expected: March 2009) ———
- Basic Skeleton library (relies on years of previous work)
- Domain-specific skeletons

  ——— (expected: Summer 2009) ———
- Streamlining, stabilising, advanced load-balancing...

SCIENCE Symbolic Computation Infrastructure for Europe

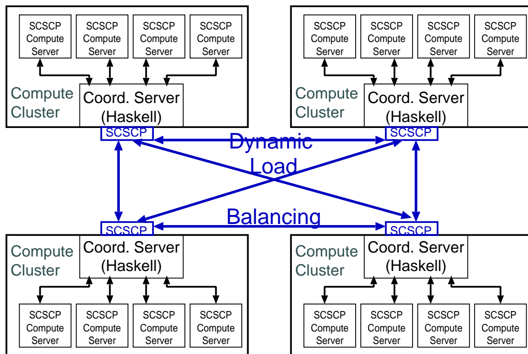
Outline

1. Introduction
2. High-Level Parallel Programming
   - Why we use functional programming
   - Algorithmic Skeletons
3. Implementation work
   - System Architecture
   - Current Status
4. Outlook and Summary

SCIENCE Symbolic Computation Infrastructure for Europe

Potential: Distributed Coordination

- Potential: **distributed middleware**



- High-level language ⇒ quick development.
- Excellent modularity ⇒ component replacement.
- High potential for automatic distributed load management.

SCIENCE Symbolic Computation Infrastructure for Europe

Summary and Outlook

- SymGrid-Par: middleware for parallelising CA systems
- Research towards
  - general and domain-specific **parallel computation schemes**
  - scalable future-proof **middleware architecture**
  - optimal performance by **dynamic automatic management**
- Parallel skeletons: User-friendly, **high-level parallelism**
- **System programming in Haskell:** type-safe, modular, large extent of automatic management and synchronisation.
- High potential for **future extensions**:
  semi-implicit (library) parallelism, advanced load balancing, ...

... which brings us straight to the next two talks:
How to write, and how to run parallel programs.

Thank you very much for your attention

Questions and Remarks

SCIENCE
Symbolic
Computation
Infrastructure for
Europe