Coordinating Data-Parallel SAC Programs with S-Net

Sven-Bodo Scholz
Clemens Grelck and Alex Shafarenko

Compiler Technology and Computer Architecture Group
University of Hertfordshire, UK

HIPS-ToPMoDRS,
26.March 2007, Long Beach

www.aether-ist.org
SAC in a nutshell

- generic array programming
- purely functional
- no-frills
- aggressive program optimisation
- sequential performance competitive with FORTRAN77
- generation of multi-threaded code without annotations

For details see www.sac-home.org
Exploitation of Data-Parallelism in **SAC**

- hinges on our data-parallel construct, the With-Loop
- all our array operations are defined by these
- granularity control by
  - array size inference
  - several fusion techniques
- architecture specific code generation
- POSIX-threads for SMPs
- taylor-made memory management
- almost linear speedups for examples such as NAS-benchmarks
So what is the catch??

Our compiler maps all concurrency into one Concurrency pattern:
So what is the catch??

Our compiler maps all concurrency into one Concurrency pattern:

if our application cannot be mapped on this pattern, we cannot exploit its concurrency!
An example: Search Algorithms
Our Approach: **S-Net**

**S-Net:**
- asynchronous component framework
- functional stream processing
- pure coordination language
- dynamic network creation
- orthogonal to the box language
**S-Net at a Glance**

- central construct: box
  wraps existing code

![Diagram of box foo]
**S-Net at a Glance**

- central construct: box wraps existing code
- connected by
  - single input stream
  - single output stream

![Diagram of box foo]

---

*Sven-Bodo Scholz*

Coordinating SAC with S-Net
S-Net at a Glance

- central construct: box wraps existing code
- connected by
  - single input stream
  - single output stream
- streams transport records: sets of named fields
  - opaque value fields
  - integer-valued tag fields
S-Net at a Glance

- central construct: box wraps existing code
- connected by
  - single input stream
  - single output stream
- streams transport records:
  sets of named fields
  - opaque value fields
  - integer-valued tag fields

box behaviour declared by type signature

```plaintext
box foo
{A,B,<T>}
{X,<T>}
{A,B,<T>}
```
S-Net at a Glance

- central construct: box wraps existing code
- connected by
  - single input stream
  - single output stream
- streams transport records:
  sets of named fields
  - opaque value fields
  - integer-valued tag fields
- box behaviour declared by type signature
- behaviour defined in box language, not S-Net

```latex
\begin{align*}
\text{box foo} & \{A,B,\langle T\rangle\} \\
& \rightarrow \{X,\langle T\rangle\}
\end{align*}
```
**S-Net at a Glance**

- Central construct: box wraps existing code
- Connected by:
  - Single input stream
  - Single output stream
- Streams transport records: sets of named fields
  - Opaque value fields
  - Integer-valued tag fields
- Box behaviour declared by type signature
- Behaviour defined in box language, not **S-Net**
- Box maps single input record to stream of output records

```plaintext
box foo
{A,B,<T>};
{X,<T>};
{X,<T>};
```
Network Combinators: Serial

box \( X := A..B \)
Network Combinators: Choice

box \( X := A \ldots B \)

- \( A \)
- \( B \)

box \( X := A \parallel B \)

- \( A \)
- \( B \)
Network Combinators: Serial Replication

- `box X := A..B`
  - A
  - B

- `box X := A||B`
  - A
  - B

- `box X := A*{<stop>}`
  - A
  - {<stop>}

Sven-Bodo Scholz

Coordinating SAC with S-Net
Network Combinators: Index Split

```plaintext
box X := A..B

box X := A||B

box X := A*{<stop>}

box X := A!<T>
```
Primitive Boxes: Filter and Sync

Filter

\{A,B,\langle T\rangle\}
\{A,\langle T\rangle,\langle X\rangle:=1\};
\{B,\langle T\rangle,\langle X\rangle:=2\}

housekeeping:

- eliminate record fields
- duplicate record fields
- add tags
- manipulate tag values
- ...

Sync

\{A,B,C,D\}
\{A,B\}
\{C,D\}

synchronisation:

- keep record that matches pattern
- all patterns matched: release merged records
- pattern already matched: pass through
Primitive Boxes: Filter and Sync

**Filter**

\[
\begin{align*}
\{A, B, <T>\} \\
\{A, <T>, <X> := 1\}; \\
\{B, <T>, <X> := 2\}
\end{align*}
\]

**Sync**

\[
\begin{align*}
\{A, B\} \\
\{C, D\} \\
\{A, B, C, D\}
\end{align*}
\]

housekeeping:
- eliminate record fields
- duplicate record fields
- add tags
- manipulate tag values
- ...

synchronisation:
- keep record that matches pattern
- all patterns matched: release merged records
- pattern already matched: pass through
Problem: Incompatible Boxes

- Box code typically designed in isolation.
- Interfaces only partially overlap.
- Network composition unfeasible in practice?
Solution: Flow Inheritance

- Any “unwanted” field is bypassed to output channel
- and is attached to any record produced in response.
Solution: Flow Inheritance

Any “unwanted” field is bypassed to output channel and is attached to any record produced in response.
Flow Inheritance: Enabling Network Combination

Flow inheritance is the key to network glue.
How to apply this to our problem?

box hor := solveOneLevel ! <k>

box vert := hor *{<done>}

box solveOneLevel
{board, opts} −> {board, opts, <k>}
| {board, <done>}

Sven-Bodo Scholz
Coordinating SAC with S-Net
How to apply this to our problem?

Sven-Bodo Scholz  
Coordinating SAC with S-Net
How to apply this to our problem?
How to apply this to our problem?
How to apply this to our problem?
How to apply this to our problem?
Conclusion

**S-Net** is

- versatile component framework
- for functional stream processing

**S-Net** features

- SISO as predominant construction principle
- asynchronous distributed computing
- dynamic network extension
- open box language interface
- few powerful combinators
Current and Future Work

▶ prototype implementation  (UH + ICL)
  ▶ runtime system
  ▶ compiler
▶ graphical tools  (UH + VTT)
  ▶ network editor
  ▶ network simulator
▶ mapping to MicroThread architecture  (UH + UvA)
  ▶ hardware support for light threads
  ▶ highly efficient synchronisation
▶ applications  (UH + VTT + others)
  ▶ network protocols
  ▶ physics simulations
  ▶ ...

Sven-Bodo Scholz
Coordinating SAC with S-Net