

# Dynamic Analysis of Data Dependencies

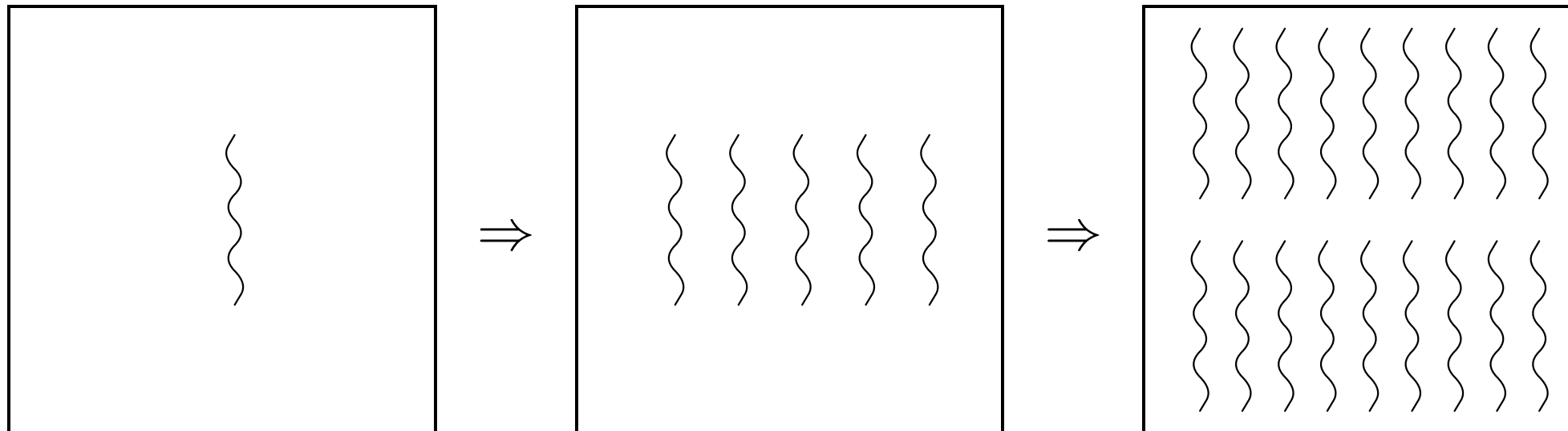
Karl-Filip Faxén

31 August 2007

# The problem

---

From single threaded program to scalable multithreading.



# Dynamic Data Dependence Analysis using Embla

---

- Support for programmers doing hand parallelisation
- Identifies program parts likely to be *independent* and thus possible to execute in parallel
- Records dependencies *dynamically* in a running program
- Results valid for *same* input, like testing
- Prototype implementation called Embla using the Valgrind instrumentation infrastructure—works for all x86 binaries under Linux

## Embla workflow: 1

---

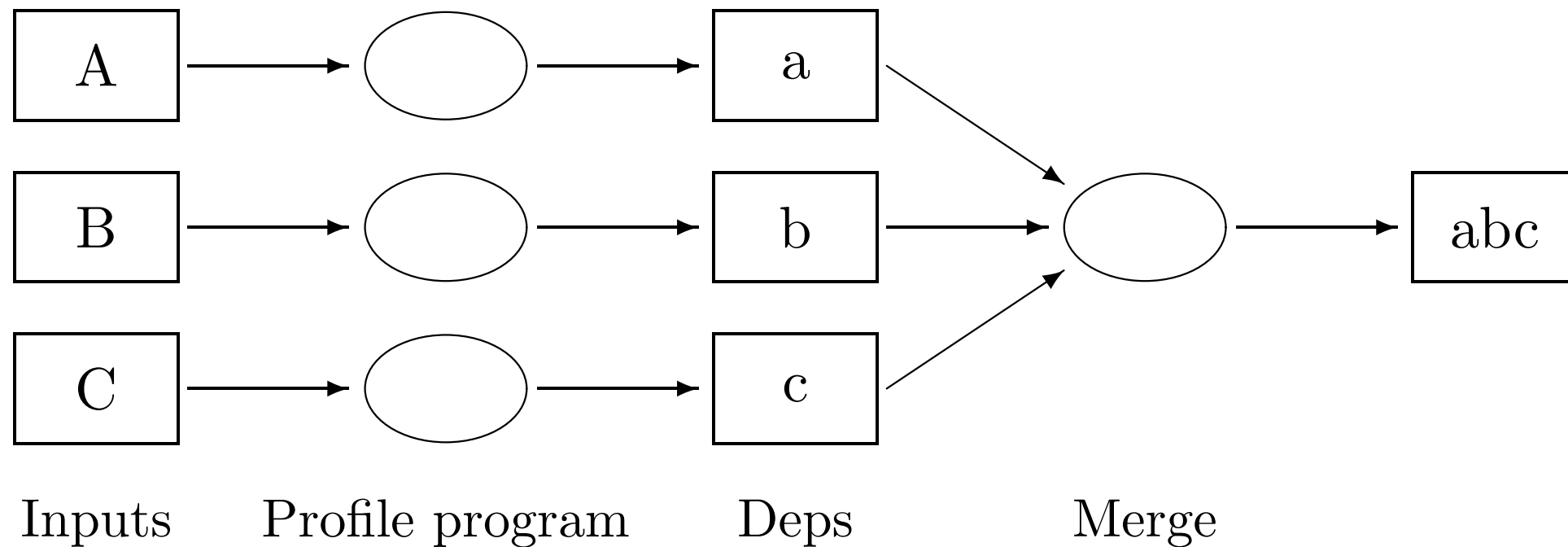
Build the program as usual.

- Embla works on the compiled program.
- Turn on debug info for interesting parts when compiling since
  - dependencies are *discovered* even in the absence of debug info,
  - but *reporting* them needs it.

## Embla workflow: 2

---

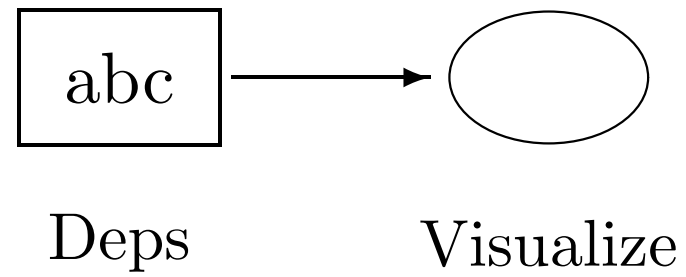
Run the program several times under Embla with different inputs and merge the profiling output.



# Embla workflow: 3

---

Visualize the collected dependencies.



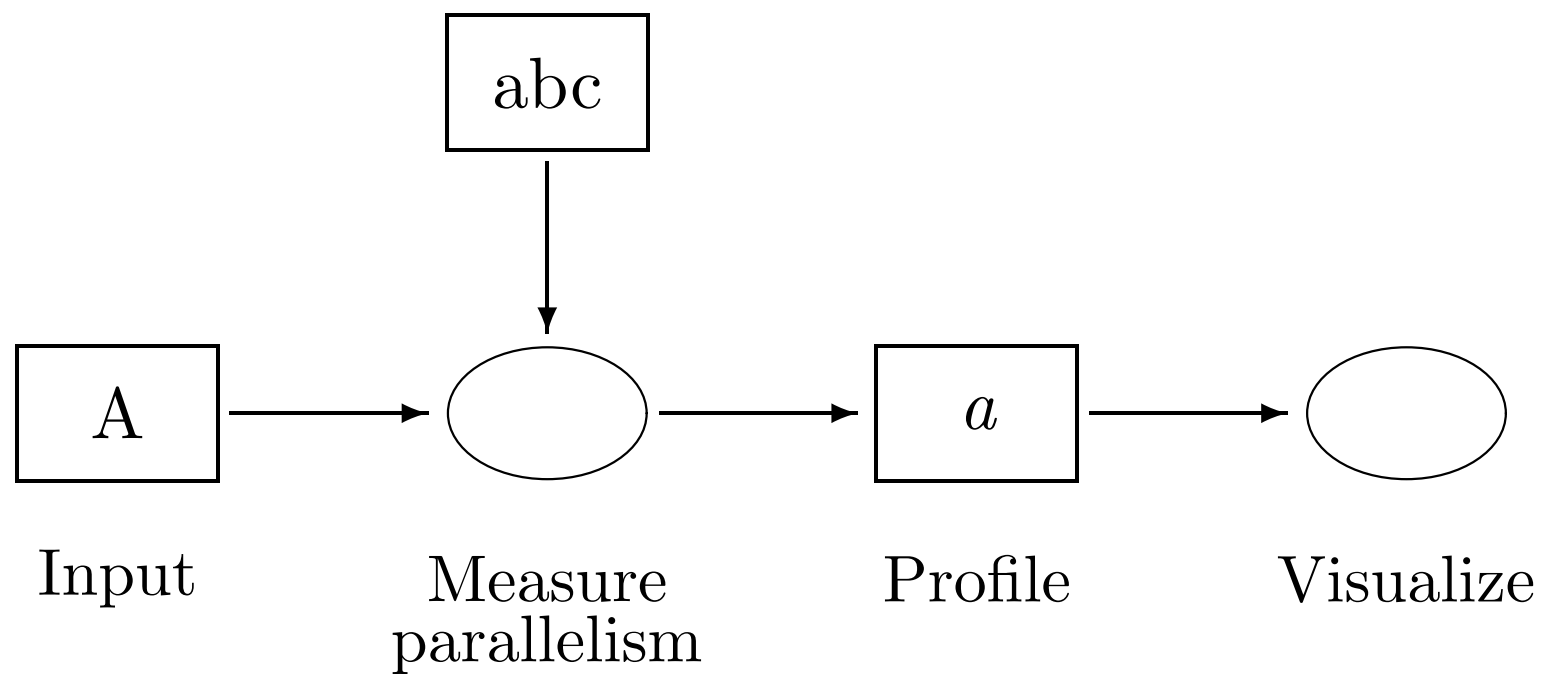


# Measure parallelism

---

Estimate parallelism implied by the dependencies found

- Shows most profitable sites to parallelize



## Embla workflow: 4

---

Transform program by adding

- explicit `pthread`s,
- lightweight threads (eg filaments),
- OpenMP pragmas
- Cilk-5 annotations (`spawn` and `sync`)
- ...

## Why *dynamic* analysis?

---

- Source code not always available for entire program (legacy, libraries).
- More than one source language might be used.
- Better precision (static analysis always conservative, finds all dependencies and then some).
- As a complement to static analysis—approximate dependencies from below as well as from above.

# Why not just insert threads by hand?

---

- Well known debugging problems with hand written threads:
  - Explosion of control state space,
  - nondeterminism (irreproducibility of bugs).
- If parallelism is introduced *systematically* based on dynamic dependence information from Embla:
  - Incorrect parallel execution only if a dependence was missed.
  - Dependencies are a feature of the sequential program.
  - Missed dependencies can be found by debugging in a deterministic, sequential environment (rerun Embla with new input).

# Dependence based parallel programming

---

1. Start with a *sequential* program.
2. Identify *independent* computations.
  - Embla: dynamic analysis (dependency profiling).
  - Dependence typing; incorrectly parallelized programs are ill-typed.
  - Static analysis as in parallelizing compilers.
3. Add minimal *annotations* specifying parallel execution:
  - Open-MP
  - ...
4. Execute in parallel with *exactly* the sequential semantics!

## Future work

---

- Analysis of threaded code
- Using Embla to parallelize real programs
- Light weight thread scheduling/execution mechanisms
  - work stealing
  - depth first parallel