Threads Study Case #2

Integral Equations

SMCC/OpCom

Threads Study Case #2

Goals of Case Study

1. Measurement of the effort required to write or convert an application.
2. Measurement of performance improvement on MP hardware.
3. Identification of techniques that can be applied to writing or converting other applications.

The Problem

- Application solves Fredholm Integral Equations of the Second kind.
- Used to model problems in heat transfer, fluid flow and vibration analysis.
- These equations look like:

$$f(x) + \lambda \int_a^b K(x, y) f(y) dy = g(x), a \le x \le b$$

- Goal is to find *f(x)*, everything else is specified.
- By using Chebyshev polynomials the equation can be transformed into a system of linear equations to be solved. The solution vector is an approximation of *f(x)*.

Threads Case Study #2

- The system of equations looks like:

$$\sum_r a_r T_r(x) + \lambda \int \left[\sum_r \sum_s b_{rs} T_r(x) T_s(y) \right] \left[\sum_r a_r T_r(y) \right] dy = \sum_r c_r T_r(x)$$

- This can be re-written as:

$$(\boldsymbol{I} + \lambda \boldsymbol{bt})\,\bar{x} = \bar{c}$$

The Next Problem

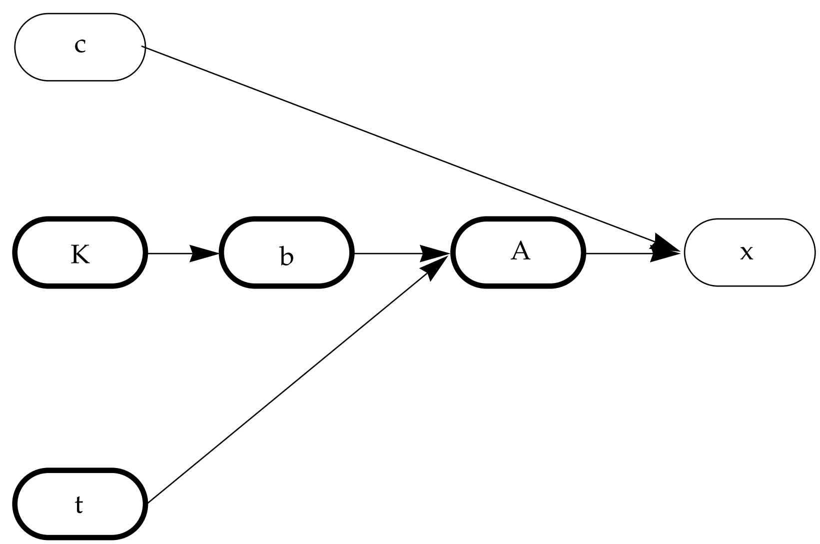
1. Computing $\boldsymbol{b}$ is an $O(N^4)$ operation (*N=100, 100 million operations*).
2. Computing $\boldsymbol{t}$ is an $O(N^2)$ operation.
3. Computing the $\boldsymbol{bt}$ matrix product is an $O(N^3)$ operation (*N=100, 1 million operations*).
4. In addition, there are three other $O(N^2)$ operations.

Threads Case Study #2

Solution

1. Build a data dependency graph.
2. Look for concurrency in the solution process.
3. Thread the parts that are concurrent.

Data Dependency Graph

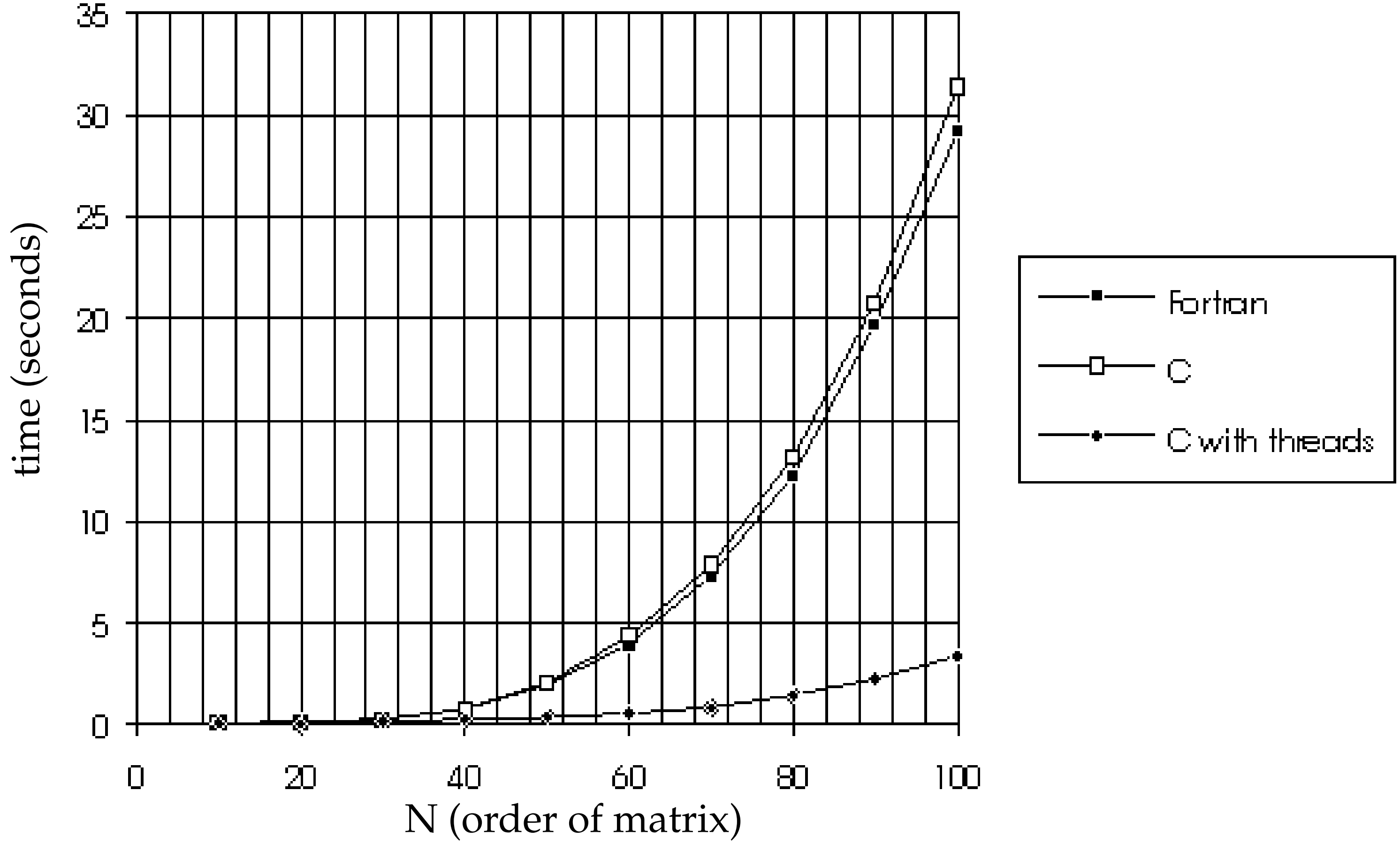


Threads Case Study #2

Results

- Tests were run on an SS1000 (8 - 50 Mhz CPUs, 512 Mb RAM).

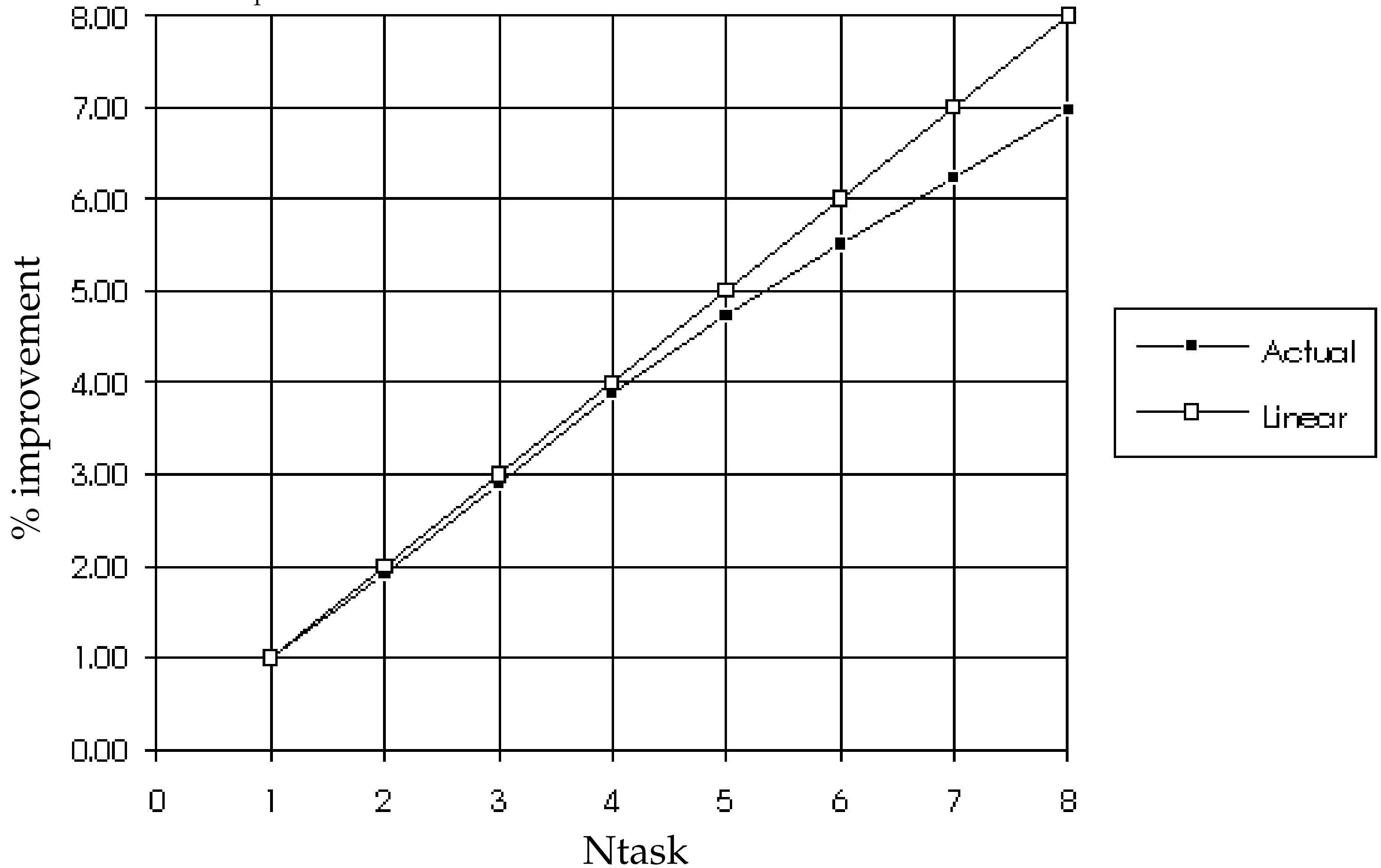
Figure 1 Order and time



Threads Case Study #2

Results (continued)

Figure 1 Ntask and % improvement



Threads Case Study #2

Conclusions

1. Conversion from C to C with threads required approximately 3 man-days of effort. Original program took 21 man-days to write and test.
2. For large values of *N*: performance improvement ranged from 5.3x to 9.2x faster (on an 8 CPU system).
3. Some learned rules:
   a. Understand the problem.
   b. Understand data generation and flow.
   c. Balance conversion effort with performance improvement goals.