

# Robust autotuning

Matteo Frigo

Intel

August 10, 2009

- Current autotuners are designed to adapt to a given machine.
- There is no such thing as “a given machine” anymore.
- Autotuners need to become **robust** and cope with dynamic variations in the environment.

## 1997:

- FFTW 1.0 out (one week hack).
- Pentium Pro 200 MHz, Ultrasparc II 166 MHz.
- Single core, mostly sequential software.
- Multiprocessor Linux uses One Big Lock.

## 2009:

- FFTW 3.2.2 out (12-year hack).
- All my machines are multi-core.
- More than 100 threads in the Linux kernel alone (disk, file system journal update, nfs, network, etc.)
- People start writing software that wants to call FFTW from multiple threads.

## Composable libraries:

- Can be called by multiple clients, in parallel.
- Without consuming too many resources (esp. memory).

## Predictions:

- Economic network effects will favor composable software.
- Noncomposable libraries will be marginalized.

# Implications of composability

## **Volatile environment:**

- There is no such thing as the “number of cores.”
- There is no such thing as the “cache size.”
- There is no such thing as the “memory bandwidth.”

## **FTW relies upon a fixed environment.**

- So does everybody else.
- We will be marginalized.

# Modeling the unknown

## Vision of a future autotuner:

- Takes **experimental measurements**.
- Also has **prior information** about the environment.
- Implements some form of inference that takes both into account.
- Heresy: at times, experimental data will be ignored in favor of the prior information.

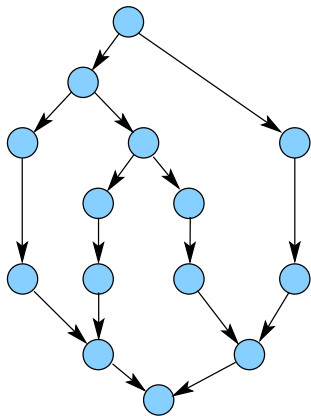
## Analogies:

- Bayesian inference.
- Kalman Filter, statistical signal processing.

## Prediction:

- We will all become masters of probabilistic reasoning.

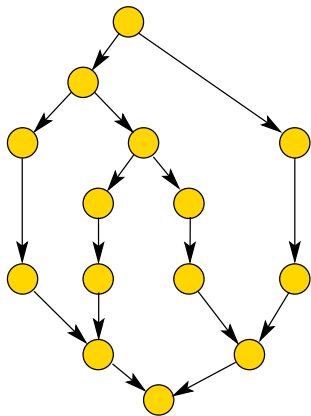
# The work and span model [Brent, Graham, Amdahl]



## Measures:

- $T_P$  = execution time on  $P$  processors

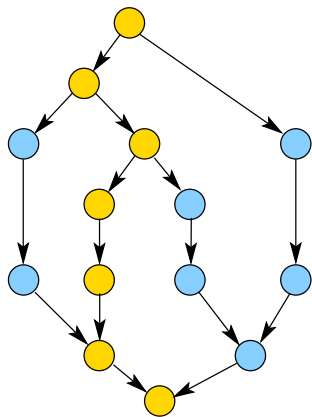
# The work and span model [Brent, Graham, Amdahl]



## Measures:

- $T_P$  = execution time on  $P$  processors
- $T_1$  = **work**

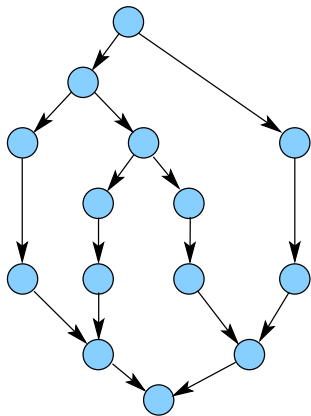
# The work and span model [Brent, Graham, Amdahl]



## Measures:

- $T_P$  = execution time on  $P$  processors
- $T_1$  = **work**
- $T_\infty$  = **span**

# The work and span model [Brent, Graham, Amdahl]



## Measures:

- $T_P$  = execution time on  $P$  processors
- $T_1$  = **work**
- $T_\infty$  = **span**

## Lower Bounds:

- $T_P \geq T_1/P$
- $T_P \geq T_\infty$

## “Easy” schedule:

- $T_P \leq T_1/P + T_\infty$
- (Optimal schedule: NP-hard.)

## Example of robust autotuner

### Choice A

- $T_1 = 40$  s
- $T_\infty = 10$  s
- $T_4 = 11$  s

### Choice B

- $T_1 = 41$  s
- $T_\infty = 1$  s
- $T_4 = 12$  s

### It may make sense to pick *B*

- Even if *A* is faster on 4 processors.
- *A* runs in  $\geq 10$  seconds, whereas *B* should run in  $\approx 41/8 + 1 \approx 6$  seconds on 8 processors.
- There is no performance counter for  $T_\infty$  — the autotuner must be explicitly instrumented.

# Summary

- There are no such things as “number of cores”, “cache size”, “memory bandwidth”, or even “pipeline”.
- Autotuners must become robust and cope with volatile environments.