Parallel & Concurrent Prog 1

no exams
homework proj. - Plaus, libraries
one term project

parallel machines

Contention  Scalability bottleneck

Symmetric Multi Processors

≤ 64 processors

NUMA

non-uniform memory access
Consistent coherent

cache coherency

cache

cc-NUMA

GRID

distributed memory

latency of communication

b/w is relatively low

master-slave model
Embarrassingly parallel — SETI@home

Folding@home

Scientific computing

— matrix mult
— FFT
— divide and conquer
— n-body problems
— SOR
  successive over-relaxation

FORTRAN

extracting ln
express ln, 2m

FORTRAN:

no recursion
no pointer

Procedures

common

a b c d e f

a = f
com = 2c
Topologies

- Bus
- Grid
- Hypercube

Libraries: PVM, MPI

PL Extensions: Message-Passing

- MPI_Send ( ... )
- MPI_Recv ( ... )

HPF
OpenMP
```c
#program omp_lines(0)
for i = 1 to 1000
    a[i] = a[i-1] * 2
```

Conventional apps: C / C++ / Java / C#

threads:

- high-performance
- low-reliability

C / C++

POSIX threads:

- `pthread_create(t, 0)`
- `pthread_join(t, result)`

Java:

- Threads
- Synchronization
- Race condition
- Nondeterminism
Darpa High Productivity Computing

Cray - Chapel
Sun - Fortress
IBM - X10

explicit parallel langs.
Java (with threads)

implicit parallel langs.
DAGs as a computation model

1 → 2 → 3 → 4

serial

1 → 4

1 → A → B → C → D

predecesor

spawn

Program: 4 nodes

\[ T_1 = \text{work} = 4 \]

\[ \frac{T_1}{P} \] optimal parallel speedup

max length

critical path

\[ T_\infty \]

\[ T_p = O\left(\frac{T_1}{P}\right) + c T_\infty \]

“Greedy schedule / Brent schedule

“Amphal’s Law”
Cilk

\begin{align*}
\text{int } fB (\text{int } v) \{ \\
& \text{ if } (v \leq 1) \{ \\
& \quad \text{return } 1; \\
& \} \text{ else } \\
& \quad \text{int } v1 = \text{Spawn } fB (v - 1); \\
& \quad \text{int } v2 = \text{Spawn } fB (v - 2); \\
& \quad \text{Sync} \\
& \quad \text{return } v1 + v2; \\
& \} \\
\end{align*}

fB(3):
\begin{align*}
& v1 = fB(2) \quad 2 \\
& v2 = fB(1) \quad 1 \\
& \text{return } 3.
\end{align*}

Work-stealing

Deque