Parallelism

\[ e ::= \ldots \mid \text{par}(e_1, e_2) \]

\[ e_1 \text{par} e_2 \mid e_1 \text{par} \langle v_1, v_2 \rangle \]

\[ \text{par}(e_1, e_2) \text{par} (v_1, v_2) \]

\[ \text{fib} \ n = \]
\[ \text{if } n \leq 1 \text{ then } \]
\[ \text{else} \]
\[ \text{let } (a, b) = \text{par}(\text{fib}(n-2), \text{fib}(n-1)) \]
\[ \text{in } a + b \]

Directed Acyclic Graphs

![Directed Acyclic Graph Diagram]

\[ \text{Work: } O(\text{fib}(n)) = O(\phi^n) \]
\[ \text{Span: } O(n) \]

**Work**: Total amount of computation
- # of nodes in DAG
- Time to run on 1 proc

**Span**: "Critical path"
- Length of longest path in DAG
- Time to run on unlimited procs

Really:

![Recursive Flow Diagram]
Schedule: Assign nodes to procs and time steps, respecting deps. Ideally, want shortest schedule - NP-Hard

\[
\begin{align*}
\text{Proc 0} & \quad \text{Proc 1} \\
1 & \quad \text{fib 3} & \quad \text{fib 1} \\
2 & \quad \text{fib 2} & \quad \text{fib 0} \\
3 & \quad l+1 & \quad \text{fib 1} \\
4 & \quad 2+1 & \\
5 & \quad 1 & \\
6 & \quad 10 & \\
\end{align*}
\]

\[\text{Proc 2}\]

Brent's Theorem: For a DAG w/ work \(W\), span \(S\), on \(P\) procs, exists a schedule of length \(\leq \frac{W}{P} + \frac{SP}{P}\).

**Pf:** Take each level at a time.

Time to execute level \(i = \left\lceil \frac{\text{# nodes at level } i}{P} \right\rceil\)

\[
\text{Time} = \sum_{i=0}^{\Sigma \# \text{levels}} \left\lceil \frac{\text{# nodes at } i}{P} \right\rceil \leq \sum \left( \frac{\text{# nodes at } i}{P} + \frac{P-1}{P} \right)
\]

\[
\leq \frac{\Sigma \text{# nodes at } i}{P} + \# \text{ levels} \left(\frac{P-1}{P}\right)
\]

\[
= \frac{W}{P} + S \frac{P-1}{P} D
\]
\[ \text{Win a factor of 2 of optimal} \]
\[ \text{Optimal: max}\ (\frac{\delta}{3}, 5) \]

Want to know \( \text{work + span} \Rightarrow \text{cost semantics} \)

\[ \begin{align*}
E_{(m,3)} & \Rightarrow \lambda x. e \ e_{(m,3)} v' \\
& \Rightarrow E_{(m,3)} v' \\
& \Rightarrow E_{(m,3)} v
\end{align*} \]

Want to know cost semantics is correct

\( \Rightarrow \) Compare to small-step “bounded implementation”

(Bellocchi + Greiner)

“Provable efficient implementation”

(Harper, Bellocchi)

Need a parallel small-step semantics

Inter-leaving

\[ \begin{align*}
E_1 & \Rightarrow E_1' \\
\text{par}\ (E_1, E_2) & \Rightarrow \text{par}\ (E_1', E_2) \\
E_2 & \Rightarrow E_2' \\
\text{par}\ (E_1, E_2) & \Rightarrow \text{par}\ (E_1, E_2') \\
\end{align*} \]

parallel

\[ \begin{align*}
E_1 & \Rightarrow E_1' \\
\text{par}\ (E_1, E_2) & \Rightarrow \text{par}\ (E_1', E_2') \\
\end{align*} \]

only counts work

\[ \begin{align*}
E_1 & \Rightarrow E_1' \\
\text{par}\ (E_1, E_2) & \Rightarrow \text{par}\ (E_1, E_2') \\
\text{par}\ (E_1, E_2) & \Rightarrow \text{par}\ (E_1', E_2') \\
E_2 & \Rightarrow E_2' \\
\text{par}\ (E_1, E_2) & \Rightarrow \text{par}\ (E_1, E_2') \\
\end{align*} \]

only counts span
Explicit threads

\(\sigma; e \Rightarrow \sigma'; e'\)

\(a_{\text{fresh}} \quad b_{\text{fresh}}\)

\(\sigma; \text{par}(e_1, e_2) \Rightarrow \sigma, a \rightarrow e_1, b \rightarrow e_2; \text{wait}(a, b)\)

\(\sigma(a) = v_1, v_1 \text{ val} \quad \sigma(b) = v_2, v_2 \text{ val}\)

..\(\sigma; \text{wait}(a, b) \Rightarrow \sigma'; (v_1, v_2)\)

\(\sigma; \text{seq}(i) \Rightarrow e_i; \sigma_i'\)

\(\sigma, a_1 \rightarrow e_1, \ldots, a_n \rightarrow e_n \Rightarrow \sigma, \sigma_1', \sigma_2', \ldots, a_n \rightarrow e_n'\)

\(\sigma_0\)

"fresh" = not used before

Can specify more about schedule