Parallelism vs. Concurrency

Key concerns

Parallelism:
Use extra resources to solve a problem faster

Concurrency:
Correctly and efficiently manage access to shared resources

May overlap

An analogy

CS1 idea: A program is like a recipe for a cook
• One cook who does one thing at a time! (Sequential)

Parallelism:
• Have lots of potatoes to slice?
• Hire helpers, hand out potatoes and knives
• But too many chefs and you spend all your time coordinating

Concurrency:
• Lots of cooks making different things, but only 4 stove burners
• Want to allow access to all 4 burners, but not cause spills or incorrect burner settings
Count Words

A.html
B.html
C.html

Count

SORT

<cow, 5>
<moo, 3>

<wombat, 15>
<cow, 23>
<the, 3>

<wombat, 15>
<purple, 3>
<the, 11>

Count

Words

Account

Info

www.bank.com

ATM

Shared-Memory Multithreading

Shared:
heap and globals

Implicit communication through sharing.

Unshared:
locals and control

Network

Message-passing, actors, "shared nothing"

Threads/processes/actors have their own private state.

Communication via explicitly sending/receiving messages.
Dataflow

Programs are DAGs.
Nodes are functions/operations.
Node executes after predecessors.

Data Parallelism

apply function/operation
to every element of array in parallel

\[
\begin{align*}
\text{do } & i=1, n \\
& z(i) = x(i) + y(i) \\
\text{enddo}
\end{align*}
\]

\[
\begin{align*}
& z(1) = 1 \\
& \text{do } i=2, n \\
& \quad z(i) = z(i-1) \times 2 \\
& \text{enddo}
\end{align*}
\]

\[
\begin{align*}
& z(1) = 1 \\
& \text{do } i=2, n \\
& \quad z(i) = z(1) \times 2^{(i-1)} \\
& \text{enddo}
\end{align*}
\]

\[
\begin{align*}
& \text{cobegin} \\
& \quad x = \text{sumRange}(0,n/2) \parallel y = \text{sumRange}(n/2,n) \\
& \text{end} \\
& z = x + y
\end{align*}
\]