Parallel Computing I

Cluster: Broadcast and Reduction

Message Passing Time Models

For `tardis`/`dr`N cluster

- Inter-node message send-time model

$$T(b) = L + \frac{1}{B} b$$

$$T(b) = 2.08 \times 10^{-4} + 1.07 \times 10^{-9} b$$

$$L = 2.08 \times 10^{-4} \text{ sec}$$
$$B = 0.935 \text{ Gbps}$$

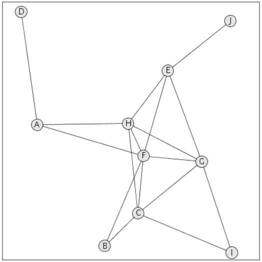
- Inter-node message scatter- and gather-time model (b bits sent to every node)

$$T(b, K) = (L + \frac{1}{B} b)(K - 1)$$

$$T(b, K) = (2.08 \times 10^{-4} + 1.07 \times 10^{-9} b)(K - 1)$$

All-Pairs Shortest-Path Problem

Given a graph with weighted edges,
determine the (length of the) shortest path between all pairs of vertices.



	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*
A	0	∞	∞	462	∞	451	∞	370	∞	∞
B	∞	0	190	∞	∞	399	∞	∞	∞	∞
C	∞	190	0	∞	∞	234	333	366	414	∞
D	462	∞	∞	0	∞	∞	∞	∞	∞	∞
E	∞	∞	∞	∞	0	359	394	269	∞	325
F	451	399	234	∞	359	0	239	144	∞	∞
G	∞	∞	333	∞	394	239	0	337	389	∞
H	370	∞	366	∞	269	144	337	0	∞	∞
I	∞	∞	414	∞	∞	∞	389	∞	0	∞
J	∞	∞	∞	∞	325	∞	∞	∞	∞	0

Floyd's All-Pairs Shortest-Path Algorithm

for $i = 0$ to $n - 1$
  for $r = 0$ to $n - 1$
    for $c = 0$ to $n - 1$
      // Update the distance from r *to* c *via* i.
      $d_{rc} \leftarrow \min(d_{rc}, d_{ri} + d_{ic})$

Floyd's All-Pairs Shortest-Path Algorithm

$$
\begin{array}{l}
\textbf{for } i = 0 \text{ to } n-1 \\
\quad \textbf{for } r = 0 \text{ to } n-1 \\
\quad\quad \textbf{for } c = 0 \text{ to } n-1 \\
\quad\quad\quad \textit{// Update the distance from } r \textit{ to } c \textit{ via } i. \\
\quad\quad\quad d_{rc} \leftarrow \min(d_{rc}, d_{ri} + d_{ic})
\end{array}
$$

How did we parallelize Floyd's Algorithm for an SMP parallel computer?

- Any sequential dependencies?
- Any load-balancing issues?

Floyd's All-Pairs Shortest-Path Algorithm

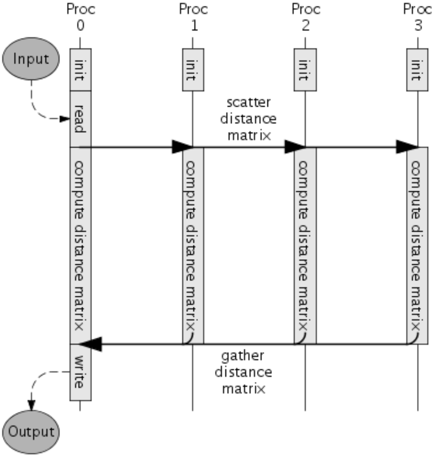
$$
\begin{array}{l}
\textbf{for } \boldsymbol{i} = \mathbf{0} \textbf{ to } \boldsymbol{n}-\mathbf{1} \\
\quad \textbf{for } \boldsymbol{r} = \mathbf{0} \textbf{ to } \boldsymbol{n}-\mathbf{1} \\
\quad\quad \textbf{for } \boldsymbol{c} = \mathbf{0} \textbf{ to } \boldsymbol{n}-\mathbf{1} \\
\quad\quad\quad \textit{// Update the distance from } \boldsymbol{r} \textit{ to } \boldsymbol{c} \textit{ via } \boldsymbol{i}. \\
\quad\quad\quad \boldsymbol{d_{rc}} \leftarrow \min(\boldsymbol{d_{rc}}, \boldsymbol{d_{ri}} + \boldsymbol{d_{ic}})
\end{array}
$$

How did we parallelize Floyd's Algorithm for an SMP parallel computer?

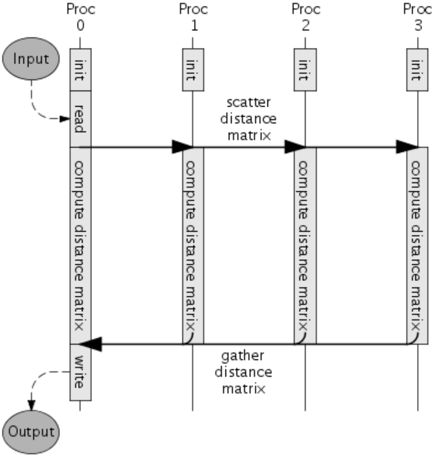
- Any sequential dependencies?
- Any load-balancing issues?

How can we parallelize Floyd's Algorithm for a cluster parallel computer?

Floyd's Algorithm on a Cluster

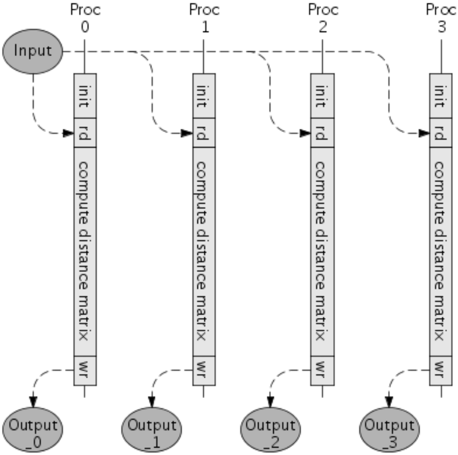


Floyd's Algorithm on a Cluster

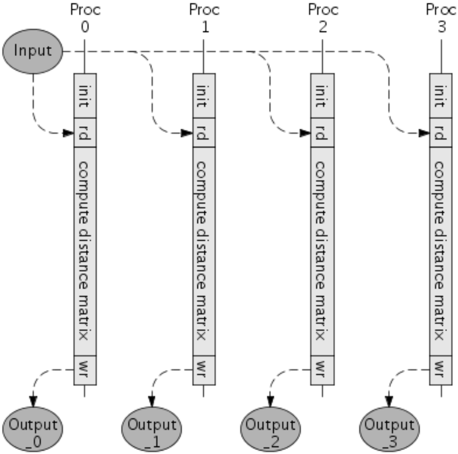


$O(n^2)$ bits scattered and gathered just for I/O.

Floyd's Algorithm on a Cluster



Floyd's Algorithm on a Cluster



Parallel input files pattern and parallel output files pattern.
If input is stored on a shared file server, then is this better than scattering?

Floyd's Algorithm on a Cluster

$$
\begin{array}{l}
\textbf{for } \boldsymbol{i} = 0 \textbf{ to } \boldsymbol{n} - 1 \\
\quad \textbf{for } \boldsymbol{r} = 0 \textbf{ to } \boldsymbol{n} - 1 \\
\quad\quad \textbf{for } \boldsymbol{c} = 0 \textbf{ to } \boldsymbol{n} - 1 \\
\quad\quad\quad \textit{// Update the distance from } \boldsymbol{r} \textit{ to } \boldsymbol{c} \textit{ via } \boldsymbol{i}. \\
\quad\quad\quad \boldsymbol{d_{rc}} \leftarrow \min(\boldsymbol{d_{rc}}, \boldsymbol{d_{ri}} + \boldsymbol{d_{ic}})
\end{array}
$$

How can we parallelize Floyd's Algorithm for a cluster parallel computer?

- What data is accessed by each process on each iteration?

Floyd's Algorithm on a Cluster

$$
\begin{array}{l}
\text{for } i = 0 \text{ to } n-1 \\
\quad \text{for } r = 0 \text{ to } n-1 \\
\quad\quad \text{for } c = 0 \text{ to } n-1 \\
\quad\quad\quad \textit{// Update the distance from } r \textit{ to } c \textit{ via } i. \\
\quad\quad\quad d_{rc} \leftarrow \min(d_{rc}, d_{ri} + d_{ic})
\end{array}
$$

How can we parallelize Floyd's Algorithm for a cluster parallel computer?

- What data is accessed by each process on each iteration?

  Every process can locally access d_{rc} and d_{ri},
  but only the process that "owns" row i can locally access d_{ic}.

Floyd's Algorithm on a Cluster

$$
\begin{array}{l}
\textbf{for } i = 0 \textbf{ to } n-1 \\
\quad \textbf{for } r = 0 \textbf{ to } n-1 \\
\quad\quad \textbf{for } c = 0 \textbf{ to } n-1 \\
\quad\quad\quad \textit{// Update the distance from } r \textit{ to } c \textit{ via } i. \\
\quad\quad\quad d_{rc} \leftarrow \min(d_{rc}, d_{ri} + d_{ic})
\end{array}
$$

How can we parallelize Floyd's Algorithm for a cluster parallel computer?

- What data is accessed by each process on each iteration?

  Every process can locally access d_{rc} and d_{ri},
  but only the process that "owns" row i can locally access d_{ic}.

  On each iteration of **for** $i = 0$ **to** $n-1$,
  one process must communicate row i to all other processes.

  Which collective communication operation?

Floyd's Algorithm on a Cluster

$$
\begin{array}{l}
\textbf{for } i = 0 \textbf{ to } n-1 \\
\quad \textbf{for } r = 0 \textbf{ to } n-1 \\
\quad\quad \textbf{for } c = 0 \textbf{ to } n-1 \\
\quad\quad\quad \textit{// Update the distance from } r \textit{ to } c \textit{ via } i. \\
\quad\quad\quad d_{rc} \leftarrow \min(d_{rc}, d_{ri} + d_{ic})
\end{array}
$$

How can we parallelize Floyd's Algorithm for a cluster parallel computer?

- What data is accessed by each process on each iteration?

  Every process can locally access d_{rc} and d_{ri},
  but only the process that "owns" row i can locally access d_{ic}.

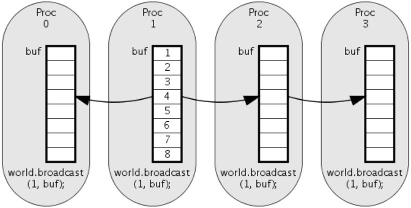
  On each iteration of **for** $i = 0$ **to** $n - 1$,
  one process must communicate row i to all other processes.

  Which collective communication operation? broadcast

Floyd's Algorithm on a Cluster

for $i = 0$ to $n-1$
  broadcast row i of d
  pfor $r = 0$ to $n-1$
    for $c = 0$ to $n-1$
      // *Update the distance from* r *to* c *via* i.
      $d_{rc} \leftarrow \min(d_{rc}, d_{ri} + d_{ic})$

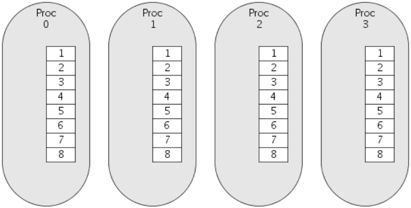
```
world.broadcast (root, buf);
```



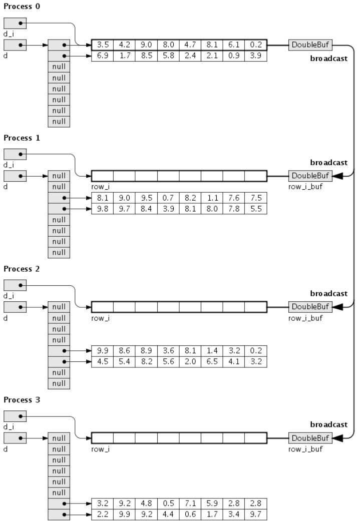
Floyd's Algorithm on a Cluster

$$
\begin{array}{l}
\textbf{for } i = 0 \textbf{ to } n-1 \\
\quad \textbf{broadcast row } i \textbf{ of } d \\
\quad \textbf{pfor } r = 0 \textbf{ to } n-1 \\
\quad\quad \textbf{for } c = 0 \textbf{ to } n-1 \\
\quad\quad\quad \textit{// Update the distance from } r \textit{ to } c \textit{ via } i. \\
\quad\quad\quad d_{rc} \leftarrow \min(d_{rc}, d_{ri} + d_{ic})
\end{array}
$$

```
world.broadcast (root, buf);
```



Floyd's Algorithm on a Cluster



FloydClu.java

code/FloydClu.java

FloydClu.java

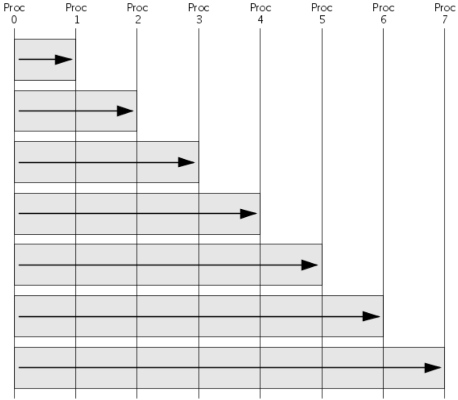
`code/FloydClu.java`

Before looking at the running-time measurements for `FloydClu`, derive a model to predict the running time (for the computation portion, omitting file I/O).

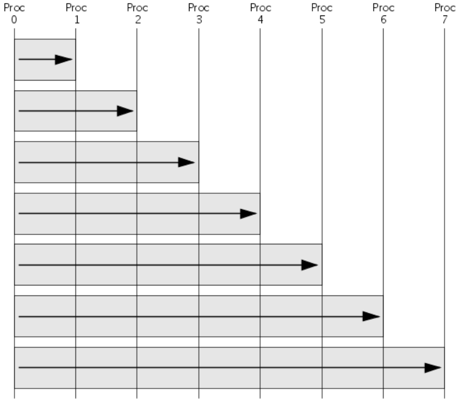
What will we need?

- a communication time model
  - an understanding of the broadcast operation's implementation
- a calculation time model

Broadcast Implementation

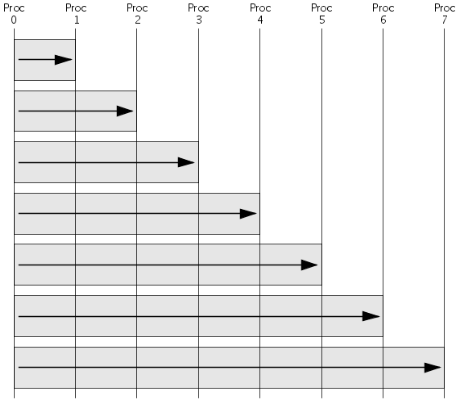


Broadcast Implementation



$$T_{\mathrm{bcast}}(b, K) = (L + \tfrac{1}{B}b)(K - 1)$$

Broadcast Implementation

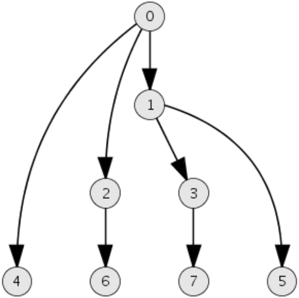


$$T_{\mathrm{bcast}}(b, K) = (L + \frac{1}{B}b)(K - 1)$$

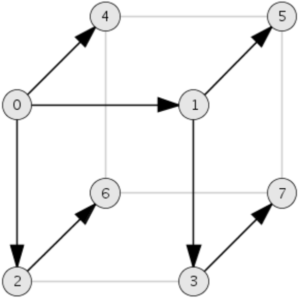
Can we do better?

Broadcast Implementation

Broadcast message pattern for $K = 8$:

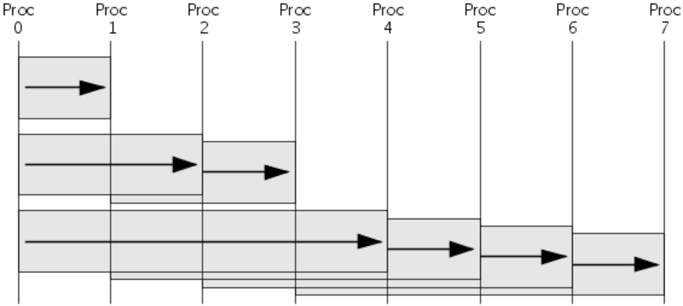


three-level tree

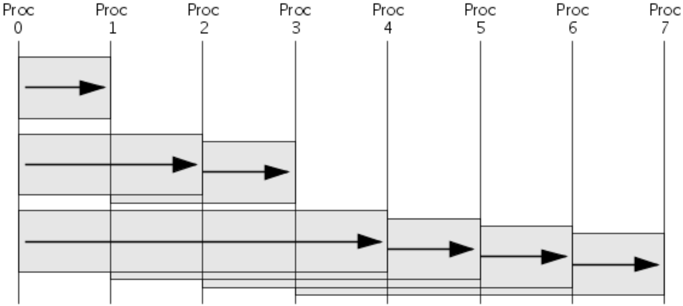


three-dimensional hypercube

Message Broadcast-Time Model

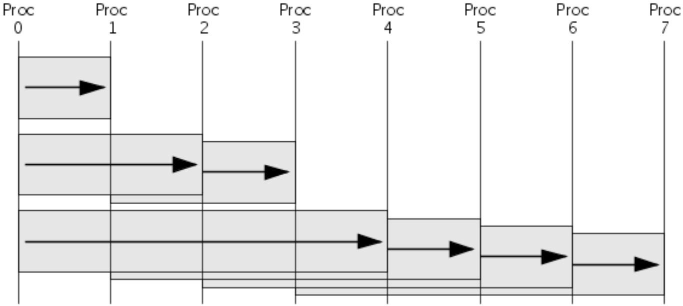


Message Broadcast-Time Model



$$T_{\mathrm{bcast}}(b, K) = (L + \frac{1}{B} b) \lceil \log_2 K \rceil$$

Message Broadcast-Time Model



$$T_{\mathrm{bcast}}(b, K) = (L + \tfrac{1}{B} b)\lceil \log_2 K \rceil$$

Aside: could/should we use this implementation for scatter/gather?

Calculation Time Model

Floyd's Algorithm is $O(n^3)$, where n is the number of vertices.

$$T_{\mathrm{calc}}^{\mathrm{Floyd}}(n, 1) = an^3$$

$$T_{\mathrm{calc}}^{\mathrm{Floyd}}(n, K) = an^3 \frac{1}{K}$$

Calculation Time Model

Measure the sequential version's running time to determine the constant.

n	Measured	Model
2000	67.942	70.889
2520	136.506	141.805
3180	269.528	284.952
4000	589.131	567.115
5040	1182.404	1134.443
6360	2474.565	2279.619

$$T_{\mathbf{calc}}^{\mathbf{Floyd}}(n, 1) = 8.86 \times 10^{-9} n^3$$

Calculation Time Model

Measure the sequential version's running time to determine the constant.

n	Measured	Model
2000	67.942	70.889
2520	136.506	141.805
3180	269.528	284.952
4000	589.131	567.115
5040	1182.404	1134.443
6360	2474.565	2279.619

$$T_{\mathrm{calc}}^{\mathrm{Floyd}}(n, 1) = 8.86 \times 10^{-9} n^3$$

$$T_{\mathrm{calc}}^{\mathrm{Floyd}}(n, K) = 8.86 \times 10^{-9} n^3 \frac{1}{K}$$

Communication Time Model

- How many broadcasts are performed?
- How many bits in each broadcast message?

Communication Time Model

- How many broadcasts are performed? n
- How many bits in each broadcast message? $64n$

Communication Time Model

- How many broadcasts are performed? **n**
- How many bits in each broadcast message? **$64n$**

$$T_{\text{comm}}^{\text{Floyd}}(n, K) = nT_{\text{bcast}}(64n, K)$$

$$T_{\text{comm}}^{\text{Floyd}}(n, K) = n(L + \tfrac{1}{B}64n)\lceil \log_2 K \rceil$$

$$T_{\text{comm}}^{\text{Floyd}}(n, K) = n(2.08 \times 10^{-4} + 6.85 \times 10^{-8} n)\lceil \log_2 K \rceil$$

Running Time Model

$$T^{\mathrm{Floyd}}(n, K) = T^{\mathrm{Floyd}}_{\mathrm{calc}}(n, K) + T^{\mathrm{Floyd}}_{\mathrm{comm}}(n, K)$$

Running Time Model

$$T^{\mathrm{Floyd}}(n, K) = T^{\mathrm{Floyd}}_{\mathrm{calc}}(n, K) + T^{\mathrm{Floyd}}_{\mathrm{comm}}(n, K)$$

$$T^{\mathrm{Floyd}}(n, K) = an^3 \frac{1}{K} + n(L + \frac{1}{B} 64n) \lceil \log_2 K \rceil$$

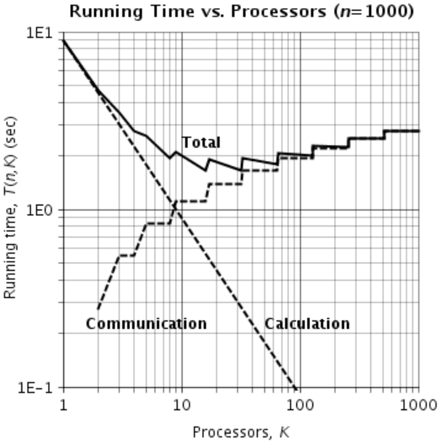
Running Time Model

$$T^{\mathrm{Floyd}}(n, K) = T^{\mathrm{Floyd}}_{\mathrm{calc}}(n, K) + T^{\mathrm{Floyd}}_{\mathrm{comm}}(n, K)$$

$$T^{\mathrm{Floyd}}(n, K) = an^3 \frac{1}{K} + n(L + \frac{1}{B} 64n) \lceil \log_2 K \rceil$$

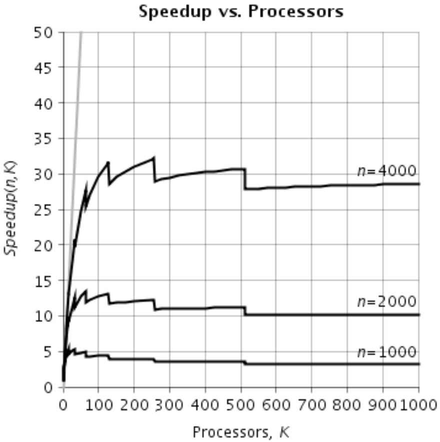
$$T^{\mathrm{Floyd}}(n, K) = 8.86 \times 10^{-9} n^3 \frac{1}{K} + n(2.08 \times 10^{-4} + 6.85 \times 10^{-8} n) \lceil \log_2 K \rceil$$

Running Time Model



$$T^{\text{Floyd}}(n, K) = 8.86\times10^{-9} n^3 \frac{1}{K} + n(2.08\times10^{-4} + 6.85\times10^{-8} n)\lceil \log_2 K \rceil$$

Speedup Model



What is the maximum speedup that the program can acheive?

Speedup Model

What is the maximum speedup that the program can acheive?

Speedup Model

What is the maximum speedup that the program can acheive?

Find the value of K that results in the minimum running time.

Speedup Model

What is the maximum speedup that the program can acheive?

Find the value of K that results in the minimum running time.

Differentiate $T^{\mathrm{Floyd}}(n, K)$ with respect to K,
set the derivative equal to 0, and solve for K.

Speedup Model

Differentiate $T^{\mathbf{Floyd}}(n, K)$ with respect to K,
set the derivative equal to 0, and solve for K.

$$\frac{d}{dK} T^{\mathbf{Floyd}}(n, K) \quad = \quad \frac{d}{dK}\left(an^3 \frac{1}{K} + n(L + \frac{1}{B} 64n) \lceil \log_2 K \rceil \right)$$

Speedup Model

Differentiate $T^{\mathrm{Floyd}}(n, K)$ with respect to K,
set the derivative equal to 0, and solve for K.

$$
\begin{aligned}
\frac{d}{dK} T^{\mathrm{Floyd}}(n, K) &= \frac{d}{dK}\left(an^3 \frac{1}{K} + n(L + \frac{1}{B}64n)\lceil \log_2 K \rceil\right) \\
&\approx \frac{d}{dK}\left(an^3 \frac{1}{K} + n(L + \frac{1}{B}64n)\log_2 K\right)
\end{aligned}
$$

Speedup Model

Differentiate $T^{\mathrm{Floyd}}(n, K)$ with respect to K,
set the derivative equal to 0, and solve for K.

$$
\begin{aligned}
\frac{d}{dK} T^{\mathrm{Floyd}}(n, K) &= \frac{d}{dK}\left(an^3 \frac{1}{K} + n(L + \frac{1}{B}64n)\lceil \log_2 K \rceil\right) \\
&\approx \frac{d}{dK}\left(an^3 \frac{1}{K} + n(L + \frac{1}{B}64n)\log_2 K\right) \\
&= \frac{d}{dK}\left(an^3 \frac{1}{K} + n(L + \frac{1}{B}64n)\frac{\ln K}{\ln 2}\right)
\end{aligned}
$$

Speedup Model

Differentiate $T^{\mathrm{Floyd}}(n, K)$ with respect to K, set the derivative equal to 0, and solve for K.

$$
\begin{aligned}
\frac{d}{dK} T^{\mathrm{Floyd}}(n, K) &= \frac{d}{dK}\left(an^3 \frac{1}{K} + n(L + \frac{1}{B}64n) \lceil \log_2 K \rceil\right) \\
&\approx \frac{d}{dK}\left(an^3 \frac{1}{K} + n(L + \frac{1}{B}64n) \log_2 K\right) \\
&= \frac{d}{dK}\left(an^3 \frac{1}{K} + n(L + \frac{1}{B}64n) \frac{\ln K}{\ln 2}\right) \\
&= -an^3 \frac{1}{K^2} + n(L + \frac{1}{B}64n) \frac{1}{K \ln 2}
\end{aligned}
$$

Speedup Model

Differentiate $T^{\mathrm{Floyd}}(n, K)$ with respect to K,
set the derivative equal to 0, and solve for K.

$$0 \;=\; -an^3\frac{1}{K^2} + n(L + \frac{1}{B}64n)\frac{1}{K\ln 2}$$

Speedup Model

Differentiate $T^{\mathbf{Floyd}}(n, K)$ with respect to K,
set the derivative equal to 0, and solve for K.

$$\begin{aligned} 0 &= -an^3\frac{1}{K^2} + n(L + \frac{1}{B}64n)\frac{1}{K\ln 2} \\ 0 &= -an^3 + n(L + \frac{1}{B}64n)\frac{1}{\ln 2}K \end{aligned}$$

Speedup Model

Differentiate $T^{\mathbf{Floyd}}(n, K)$ with respect to K,
set the derivative equal to 0, and solve for K.

$$
\begin{aligned}
0 &= -an^3 \frac{1}{K^2} + n(L + \frac{1}{B} 64n) \frac{1}{K \ln 2} \\
0 &= -an^3 + n(L + \frac{1}{B} 64n) \frac{1}{\ln 2} K \\
an^3 \ln 2 &= n(L + \frac{1}{B} 64n) K
\end{aligned}
$$

Speedup Model

Differentiate $T^{\mathrm{Floyd}}(n, K)$ with respect to K,
set the derivative equal to 0, and solve for K.

$$\begin{aligned}
0 &= -an^3\frac{1}{K^2} + n(L + \tfrac{1}{B}64n)\frac{1}{K\ln 2} \\
0 &= -an^3 + n(L + \tfrac{1}{B}64n)\frac{1}{\ln 2}K \\
an^3\ln 2 &= n(L + \tfrac{1}{B}64n)K \\
K &= \frac{an^2\ln 2}{L + \frac{1}{B}64n}
\end{aligned}$$

Speedup Model

Differentiate $T^{\mathrm{Floyd}}(n, K)$ with respect to K,
set the derivative equal to 0, and solve for K.

$$
\begin{aligned}
0 &= -an^3 \frac{1}{K^2} + n(L + \frac{1}{B}64n)\frac{1}{K \ln 2} \\
0 &= -an^3 + n(L + \frac{1}{B}64n)\frac{1}{\ln 2}K \\
an^3 \ln 2 &= n(L + \frac{1}{B}64n)K \\
K &= \frac{an^2 \ln 2}{L + \frac{1}{B}64n}
\end{aligned}
$$

$$
K^{\mathrm{Floyd}}_{\mathrm{best}} = \frac{an^2 \ln 2}{L + \frac{1}{B}64n}
$$

Speedup Model

$$K_{\text{best}}^{\text{Floyd}}(n) = \frac{an^2 \ln 2}{L + \frac{1}{B} 64n}$$

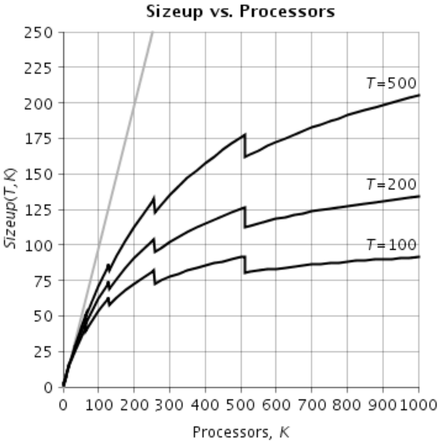
$$K_{\text{best}}^{\text{Floyd}}(n) = \frac{6.14 \times 10^{-9} n^2}{2.08 \times 10^{-4} + 6.85 \times 10^{-8} n}$$

$$K_{\text{best}}^{\text{Floyd}}(1000) \approx 22$$

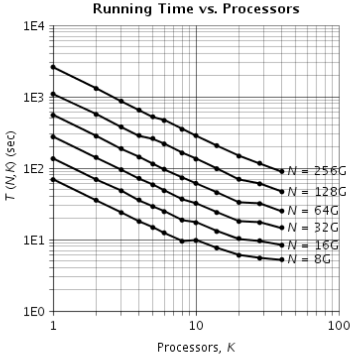
$$Speedup^{\text{Floyd}}(1000, 22) = 4.963$$

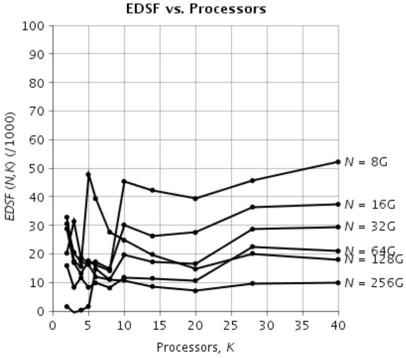
$$Eff^{\text{Floyd}}(1000, 22) = 0.226$$

Sizeup Model

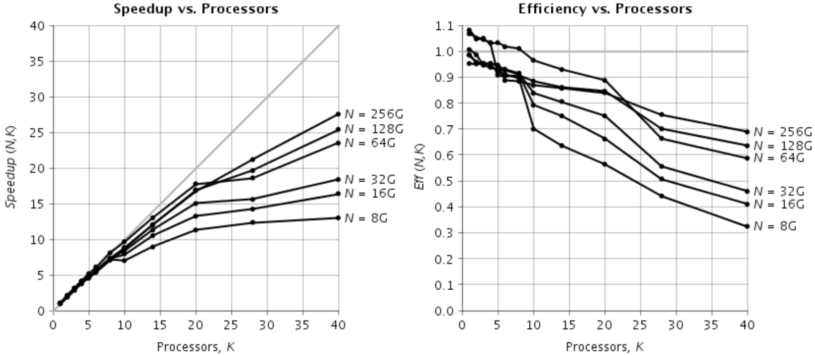


FloydClu Running Time and EDSF

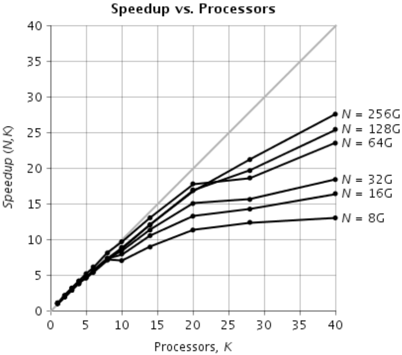


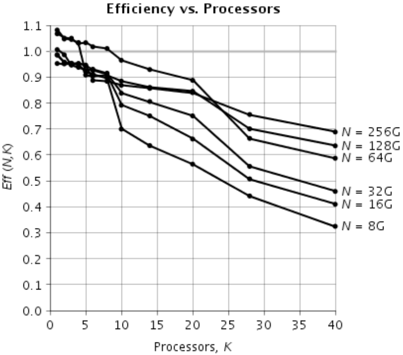


FloydClu Speedup and Efficiency



FloydClu Speedup and Efficiency



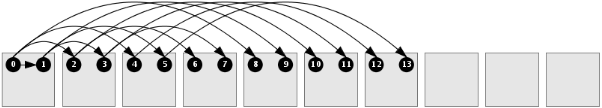


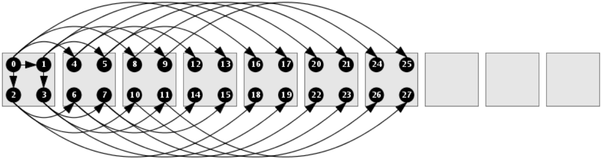
How well does the running time model predict the actual running time?

`FloydClu` Experimental Results

Recall that the `tardis`/`dr`N cluster
is actually a cluster of SMP backend nodes.

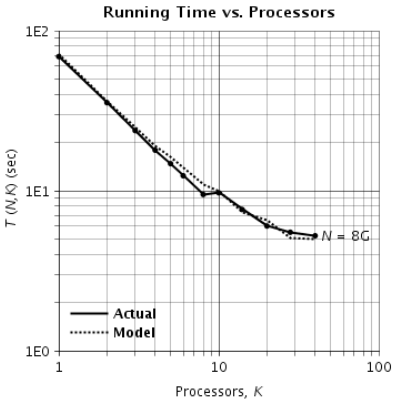
Some broadcast messages are inter-node and some are intra-node:





Slightly different $T_{\mathbf{bcast}}(b, K)$ terms depending on K.

FloydClu Experimental Results



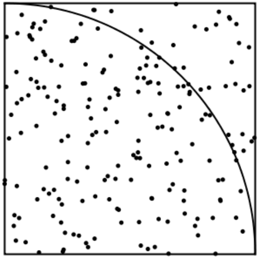
`FloydClu`

To get good parallel performance,
program must have much more computation than communication.

Unfortunately, `FloydClu` does not.

Estimating π

Suppose we have a square dartboard with sides of length $\mathbf{1m}$ and with a quarter-circle of radius $\mathbf{1m}$ inscribed.



- Let N be the number of darts thrown.
- Let C be the number of darts that landed within the q-circle.

$$\frac{C}{N} \approx \frac{\frac{1}{4} \cdot \pi \cdot (1\mathrm{m})^2}{(1\mathrm{m})^2} = \frac{\pi}{4}$$

Estimating π

SMP parallel program

- shared global PRNG or per-thread PRNGs
- shared global counter or per-thread counters

Estimating π

SMP parallel program

- shared global PRNG or per-thread PRNGs
- shared global counter or per-thread counters

Cluster parallel program

- shared global PRNG or per-thread PRNGs
- shared global counter or per-thread counters

Estimating π

SMP parallel program

- shared global PRNG or per-thread PRNGs
- shared global counter or per-thread counters

Cluster parallel program

- shared global PRNG or per-thread PRNGs
- shared global counter or per-thread counters

Why is the "wrong" choice disasterous on a cluster?

Estimating π

SMP parallel program

- shared global PRNG or per-thread PRNGs
- shared global counter or per-thread counters

Cluster parallel program

- shared global PRNG or per-thread PRNGs
- shared global counter or per-thread counters

Why is the "wrong" choice disasterous on a cluster?

To implement the "right" choice,
which collective communication operation?

Estimating π

SMP parallel program

- shared global PRNG or per-thread PRNGs
- shared global counter or per-thread counters

Cluster parallel program

- shared global PRNG or per-thread PRNGs
- shared global counter or per-thread counters

Why is the "wrong" choice disasterous on a cluster?

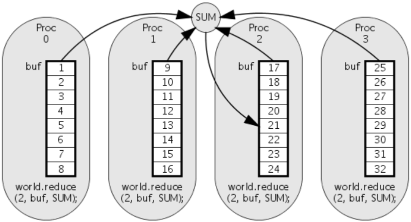
To implement the "right" choice,
which collective communication operation? reduce

Monte Carlo Pi on a Cluster

```
world.reduce (root, buf, op);
```

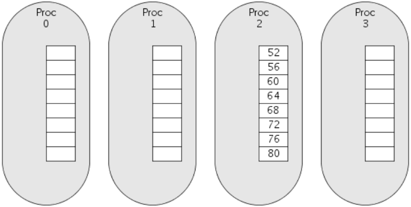
Monte Carlo Pi on a Cluster

```
world.reduce (root, buf, op);
```



Monte Carlo Pi on a Cluster

```
world.reduce (root, buf, op);
```



PiClu.java

code/PiClu.java

`PiClu.java`

`code/PiClu.java`

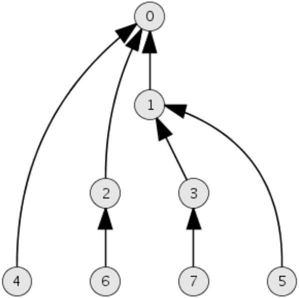
Before looking at the running-time measurements for `PiClu`, derive a model to predict the running time.

What will we need?

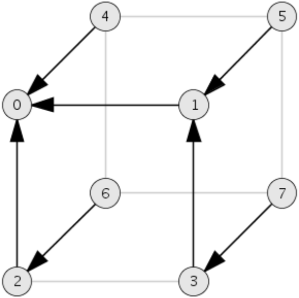
- a communication time model
  - an understanding of the reduce operation's implementation
- a calculation time model

Reduce Implementation

Reduce message pattern for $K = 8$:

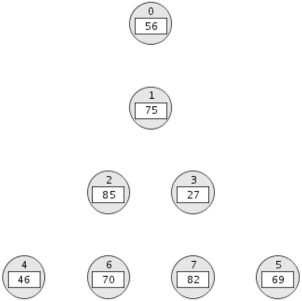


three-level tree

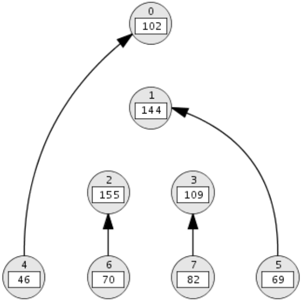


three-dimensional hypercube

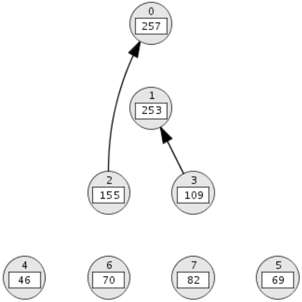
Reduce Implementation



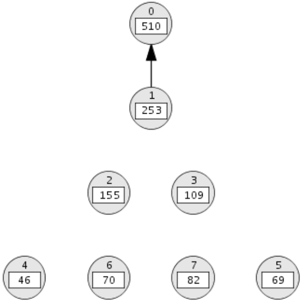
Reduce Implementation



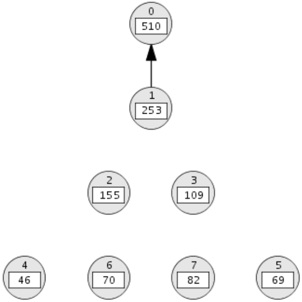
Reduce Implementation



Reduce Implementation

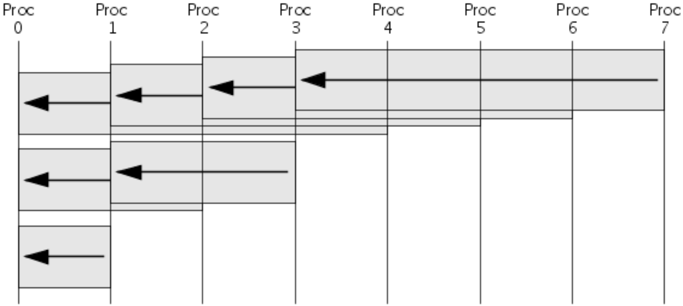


Reduce Implementation

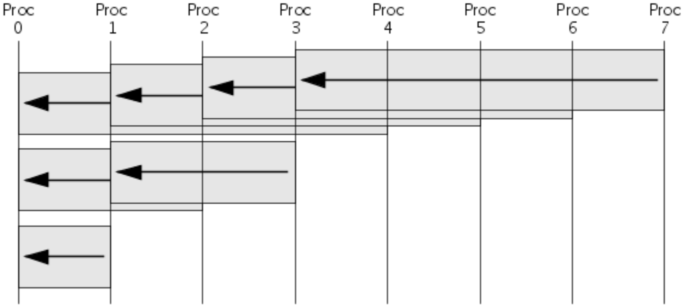


What can non-root processes assume about their data buffer after performing a `reduce` operation?

Message Reduce-Time Model



Message Reduce-Time Model



$$T_{\mathrm{reduce}}(b, K) = (L + \frac{1}{B}b)\lceil \log_2 K \rceil$$

Calculation Time Model

Monte-Carlo Pi algorithm is $O(N)$, where N is the number of points.

$$T^{\mathrm{Pi}}_{\mathrm{calc}}(N,1) = aN$$

$$T^{\mathrm{Pi}}_{\mathrm{calc}}(N,K) = aN\frac{1}{K}$$

$$T^{\mathrm{Pi}}_{\mathrm{calc}}(N,1) = 6.71\times 10^{-8}N$$

$$T^{\mathrm{Pi}}_{\mathrm{calc}}(N,K) = 6.71\times 10^{-8}N\frac{1}{K}$$

Communication Time Model

- How many reduces are performed?
- How many bits in each reduce message?

Communication Time Model

- How many reduces are performed? **1**
- How many bits in each reduce message? **64**

Communication Time Model

- How many reduces are performed? **1**
- How many bits in each reduce message? **64**

$$T^{\mathrm{Pi}}_{\mathrm{comm}}(N, K) = 1\,T_{\mathrm{reduce}}(64, K)$$

$$T^{\mathrm{Pi}}_{\mathrm{comm}}(N, K) = (L + \tfrac{1}{B}64)\lceil \log_2 K \rceil$$

$$T^{\mathrm{Pi}}_{\mathrm{comm}}(N, K) = (2.08\times 10^{-4} + 6.85\times 10^{-8})\lceil \log_2 K \rceil$$

Running Time Model

$$T^{\mathrm{Pi}}(N,K) = T^{\mathrm{Pi}}_{\mathrm{calc}}(N,K) + T^{\mathrm{Pi}}_{\mathrm{comm}}(N,K)$$

$$T^{\mathrm{Pi}}(N,K) = aN\frac{1}{K} + (L + \frac{1}{B}64)\lceil \log_2 K \rceil$$

$$T^{\mathrm{Pi}}(N,K) = 6.71\times 10^{-8} N\frac{1}{K} + (2.08\times 10^{-4} + 6.85\times 10^{-8})\lceil \log_2 K \rceil$$

Running Time Model

$$T^{\mathrm{Pi}}(N,K) = T^{\mathrm{Pi}}_{\mathrm{calc}}(N,K) + T^{\mathrm{Pi}}_{\mathrm{comm}}(N,K)$$

$$T^{\mathrm{Pi}}(N,K) = aN\frac{1}{K} + (L + \frac{1}{B}64)\lceil \log_2 K \rceil$$

$$T^{\mathrm{Pi}}(N,K) = 6.71\times 10^{-8} N\frac{1}{K} + (2.08\times 10^{-4} + 6.85\times 10^{-8})\lceil \log_2 K \rceil$$

Same "problem" as Floyd's Algorithm?

- calculation time decreases as a function of K
- communication time increases as a function of K

Speedup Model

What is the maximum speedup that the program can acheive?

$$
\begin{aligned}
\frac{d}{dK} T^{\mathrm{Pi}}(N,K) &= \frac{d}{dK}\left(aN\frac{1}{K} + (L + \frac{1}{B}64)\lceil \log_2 K \rceil\right) \\
&\approx \frac{d}{dK}\left(aN\frac{1}{K} + (L + \frac{1}{B}64)\log_2 K\right) \\
&= \frac{d}{dK}\left(aN\frac{1}{K} + (L + \frac{1}{B}64)\frac{\ln K}{\ln 2}\right) \\
&= -aN\frac{1}{K^2} + (L + \frac{1}{B}64)\frac{1}{K \ln 2}
\end{aligned}
$$

$$
\begin{aligned}
0 &= -aN\frac{1}{K^2} + (L + \frac{1}{B}64)\frac{1}{K \ln 2} \\
0 &= -aN + (L + \frac{1}{B}64)\frac{1}{\ln 2}K \\
aN \ln 2 &= (L + \frac{1}{B}64)K \\
K &= \frac{aN \ln 2}{L + \frac{1}{B}64}
\end{aligned}
$$

$$
K^{\mathrm{Pi}}_{\mathrm{best}}(N) = \frac{aN \ln 2}{L + \frac{1}{B}64}
$$

Speedup Model

$$K^{\mathrm{Pi}}_{\mathrm{best}}(N) = \frac{aN \ln 2}{L + \frac{1}{B} 64}$$

$$K^{\mathrm{Pi}}_{\mathrm{best}}(N) = \frac{4.65 \times 10^{-8} N}{2.08 \times 10^{-4} + 6.85 \times 10^{-8}} = 2.24 \times 10^{-4} N$$

$$K^{\mathrm{Pi}}_{\mathrm{best}}(1 \times 10^{9}) \approx 224000$$

$$\mathit{Speedup}^{\mathrm{Pi}}(1 \times 10^{9}, 224000) = 16589.256$$

$$\mathit{Eff}^{\mathrm{Pi}}(1 \times 10^{9}, 224000) = 0.074$$

Speedup Model

$$K^{\mathrm{Floyd}}_{\mathrm{best}}(1000) \approx 22$$

$$\mathit{Speedup}^{\mathrm{Floyd}}(1000, 22) = 4.963$$

$$K^{\mathrm{Pi}}_{\mathrm{best}}(1 \times 10^{9}) \approx 224000$$

$$\mathit{Speedup}^{\mathrm{Pi}}(1 \times 10^{9}, 224000) = 16589.256$$

Speedup Model

$$K_{\text{best}}^{\text{Floyd}}(1000) \approx 22$$

$$Speedup^{\text{Floyd}}(1000, 22) = 4.963$$

$$K_{\text{best}}^{\text{Pi}}(1 \times 10^9) \approx 224000$$

$$Speedup^{\text{Pi}}(1 \times 10^9, 224000) = 16589.256$$

Intuitively, why is `PiClu` better than `FloydClu`?

Speedup Model

$$K_{\mathrm{best}}^{\mathrm{Floyd}}(1000) \approx 22$$

$$Speedup^{\mathrm{Floyd}}(1000, 22) = 4.963$$

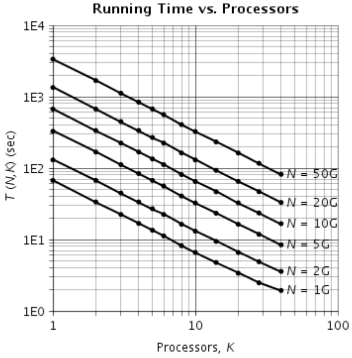
$$K_{\mathrm{best}}^{\mathrm{Pi}}(1\times 10^9) \approx 224000$$

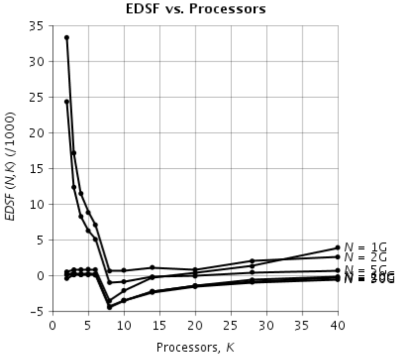
$$Speedup^{\mathrm{Pi}}(1\times 10^9, 224000) = 16589.256$$

Intuitively, why is `PiClu` better than `FloydClu`?

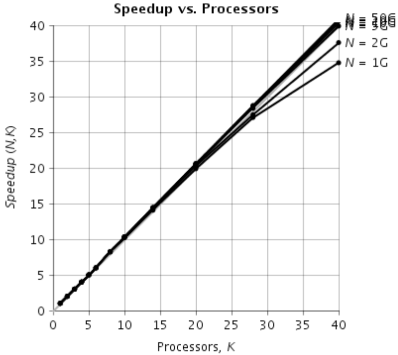
- `FloydClu`: $O(n^3/K)$ calculation, $O(n^2 \log_2 K)$ communication
- `PiClu`: $O(N/K)$ calculation, $O(\log_2 K)$ communication

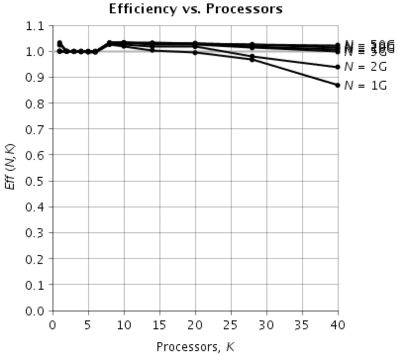
PiClu Running Time and EDSF





PiClu Speedup and Efficiency

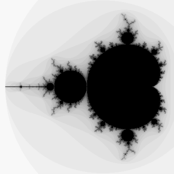


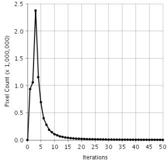


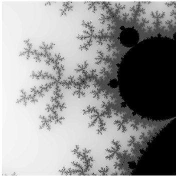
Histogram of the Mandelbrot Set

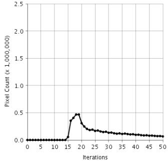
Compute the number of pixels whose iteration count was 0, 1, . . . , L.

- Number of pixels whose iteration count was L estimates the area of the Mandelbrot set.









Mandelbrot Histogram on a Cluster

- How to balance load in a cluster parallel program?

Mandelbrot Histogram on a Cluster

- How to balance load in a cluster parallel program?
  master-worker pattern

Mandelbrot Histogram on a Cluster

- How to balance load in a cluster parallel program?
  master-worker pattern

- What messages sent from master-to-worker? from worker-to-master?

Mandelbrot Histogram on a Cluster

- How to balance load in a cluster parallel program?
  master-worker pattern

- What messages sent from master-to-worker? from worker-to-master?
  - message sent to a worker containing a range
  - message sent to the master containing histogram data

  or

  - message sent to a worker containing a range
  - message sent to the master containing nothing
  - message sent to the master containing histogram data

  What is the advantage of the second approach?

MSHistogramClu.java

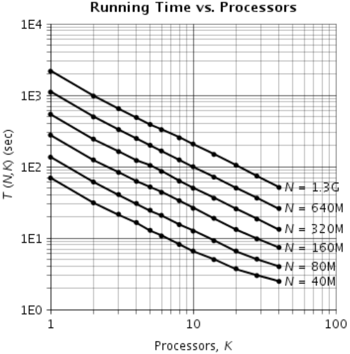
code/MSHistogramClu.java

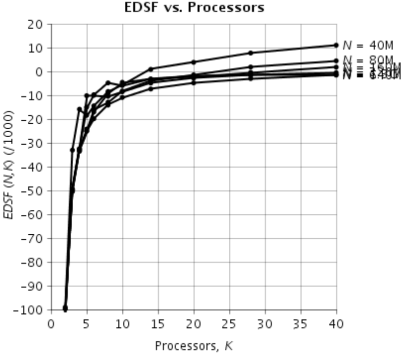
MSHistogramClu.java

code/MSHistogramClu.java

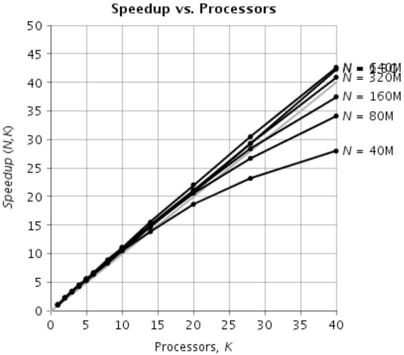
Are message tags necessary in this program?

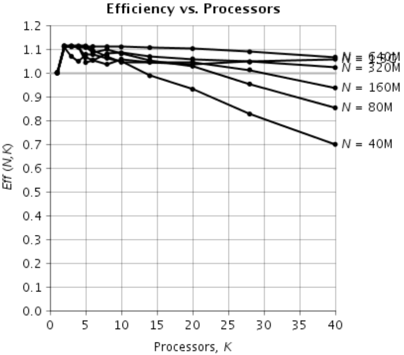
MSHistogramClu Running Time and EDSF





MSHistogramClu Speedup and Efficiency





`Worker*` Classes

Parallel Java has added support for master-worker pattern:

- `WorkerTeam`
- `WorkerRegion`
- `WorkerIntegerForLoop`
- `WorkerLongForLoop`

WorkerTeam

- `WorkerTeam()`
  Construct a new worker team with one thread per process and using the world communicator for message passing.
- `WorkerTeam(Comm comm)`
  Construct a new worker team with one thread per process and using the given communicator for message passing.
- `void execute(WorkerRegion theRegion)`
  Execute the given worker region.

Note: Every process in the cluster program creates a `WorkerTeam`, and every process in the cluster program calls `execute`.

A master thread and a worker thread will be created in the rank-0 process.

WorkerRegion

- **void** start()
  Perform initialization actions before parallel execution begins.
- **abstract void** run()
  Execute parallel code.
- **void** execute(**int** first, **int** last, WorkerIntegerForLoop theLoop)
  Execute a worker for loop within this worker region.
- **void** finish()
  Perform finalization actions after parallel execution ends.

Note: Every process in the cluster program calls `start()` and `finish()`.

WorkerIntegerForLoop

- `IntegerSchedule schedule()`
  Determine this worker for loop's schedule.
- `void start()`
  Perform per-worker per-thread initialization actions before starting the loop iterations.
- `abstract void run(int first, int last)`
  Execute one chunk of iterations of this worker for loop.
- `void finish()`
  Perform per-worker per-thread finalization actions after finishing the loop iterations.

`WorkerIntegerForLoop`

- `IntegerSchedule schedule()`
  Determine this worker for loop's schedule.
- **`void`** `start()`
  Perform per-worker per-thread initialization actions before starting the loop iterations.
- **`abstract void`** `run(`**`int`** `first,` **`int`** `last)`
  Execute one chunk of iterations of this worker for loop.
- **`void`** `finish()`
  Perform per-worker per-thread finalization actions after finishing the loop iterations.

Any functionality is missing with respect to `ParallelTeam`/`ParallelRegion`/`ParallelForLoop`?

`WorkerIntegerForLoop`

- `IntegerSchedule schedule()`
  Determine this worker for loop's schedule.
- **`void`** `start()`
  Perform per-worker per-thread initialization actions before starting the loop iterations.
- **`abstract void`** `run(`**`int`** `first,` **`int`** `last)`
  Execute one chunk of iterations of this worker for loop.
- **`void`** `finish()`
  Perform per-worker per-thread finalization actions after finishing the loop iterations.

Any functionality is missing with respect to `ParallelTeam`/`ParallelRegion`/`ParallelForLoop`?
Opportunity to communicate task-specific data between master and worker; happens implicitly via shared memory in SMP parallel program.

WorkerIntegerForLoop

- `void sendTaskInput(Range range, Comm comm, int wRank, int tag)`
  Send additional input data associated with a task. (master)
- `void receiveTaskInput(Range range, Comm comm, int mRank, int tag)`
  Receive additional input data associated with a task. (worker)
- `void sendTaskOutput(Range range, Comm comm, int mRank, int tag)`
  Send additional output data associated with a task. (worker)
- `void receiveTaskOutput(Range range, Comm comm, int wRank, int tag)`
  Receive additional output data associated with a task. (master)

MSHistogramCluNew.java and MandelbrotSetClu2New.java

code/MSHistogramCluNew.java
code/MandelbrotSetClu2New.java