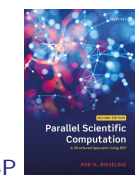
Parallel algorithm for hybrid-BSP

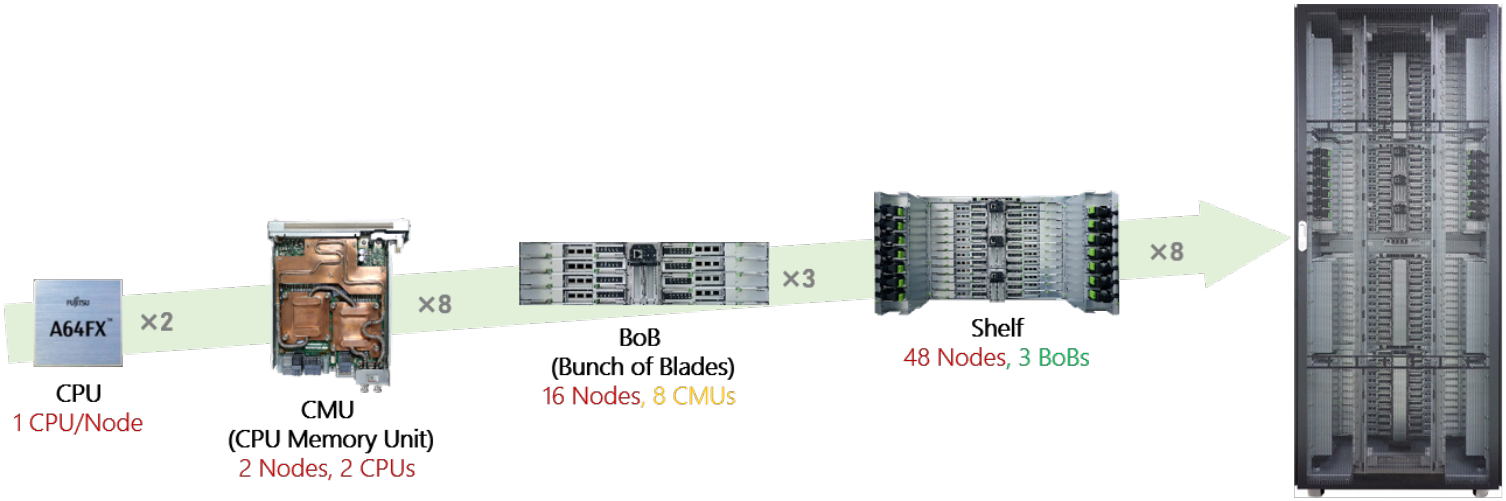
Section 4.10 of Parallel Scientific Computation, 2nd edition

Rob H. Bisseling

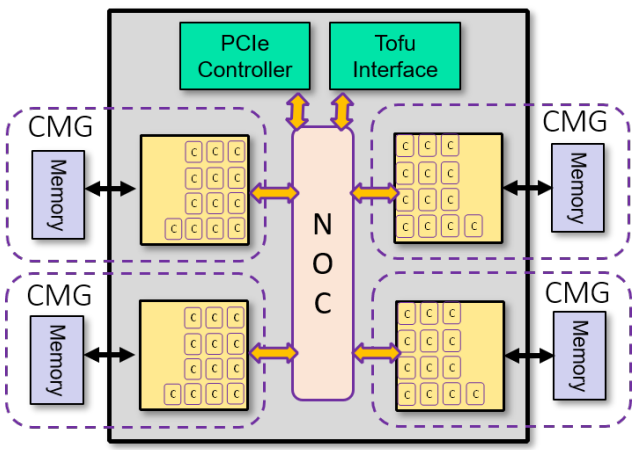
Utrecht University

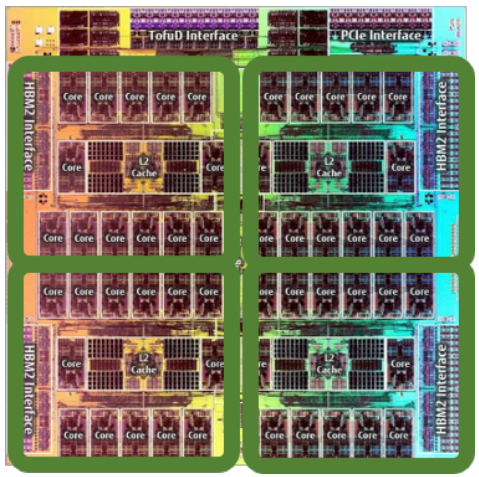


Hierarchical architecture of the Fugaku supercomputer

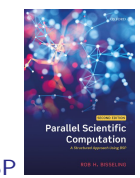


Courtesy of FUJITSU LIMITED

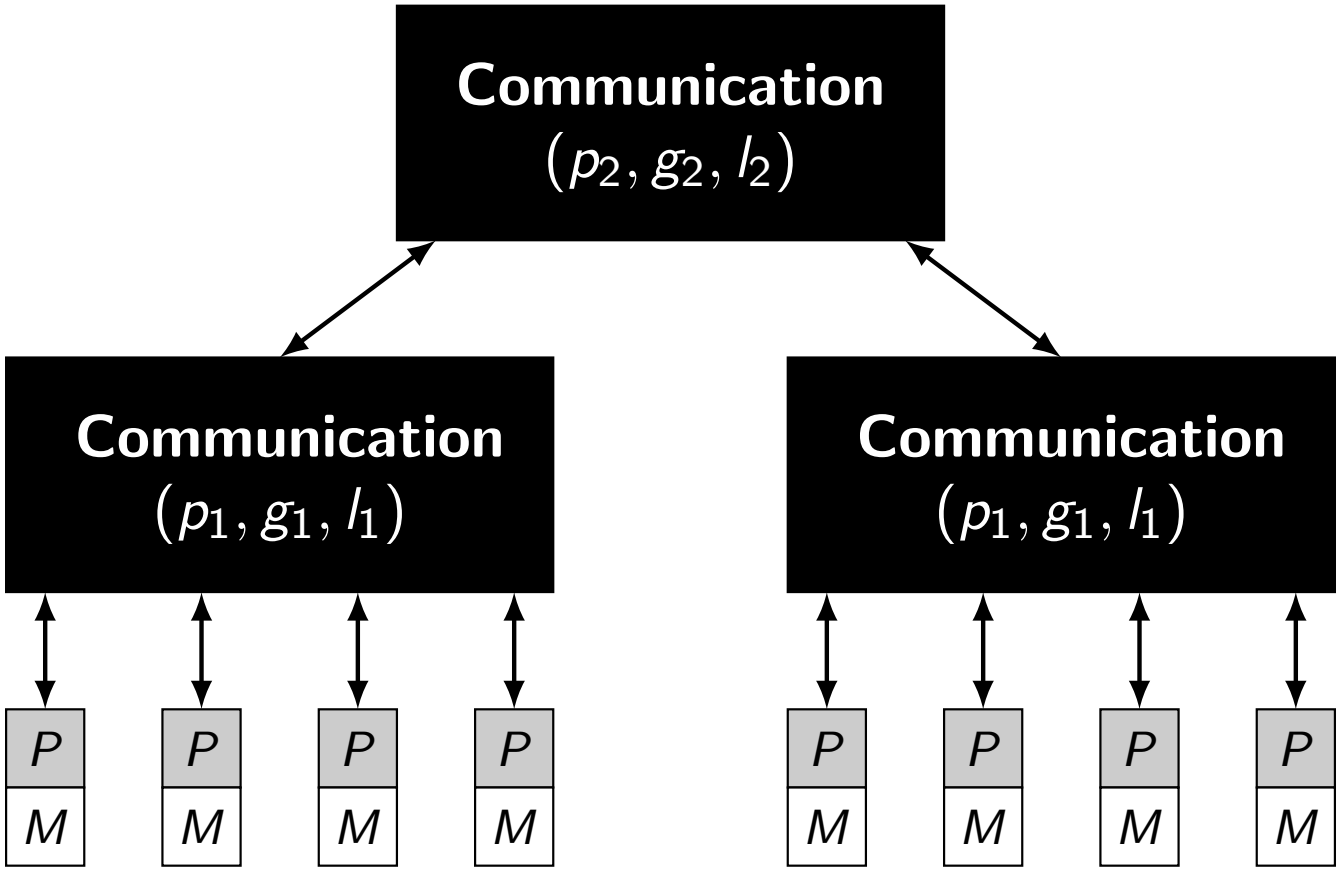




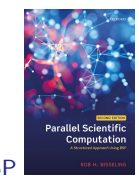
- Hybrid shared/distributed-memory computer.
- 1 node $= 4 \times 12$ compute cores.



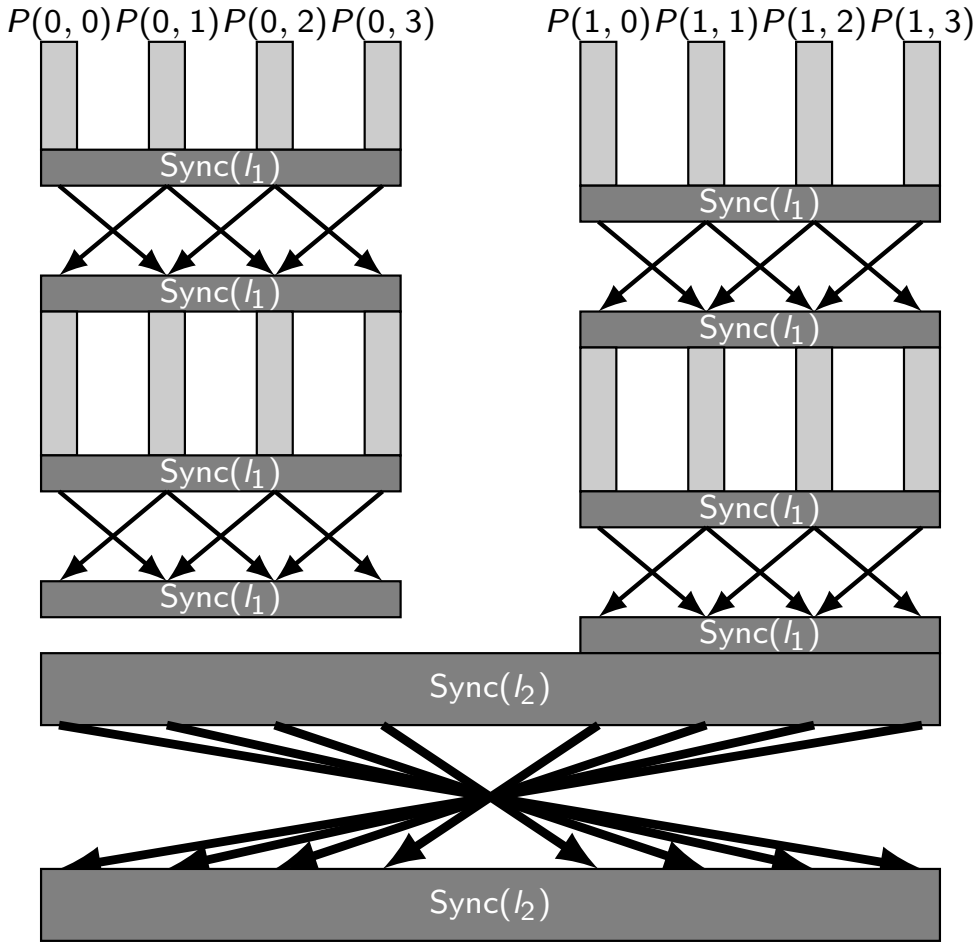
Architecture of a hybrid-BSP computer



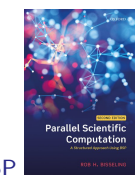
- $p_2 = 2$ nodes with $p_1 = 4$ cores per node.
- Assumption: $g_1 \ll g_2$ and $l_1 \ll l_2$.



Superstep structure of a hybrid-BSP algorithm



- The algorithm has global supersteps involving all processors and local supersteps carried out within a node.

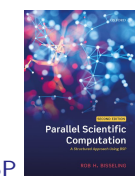


Hybrid-BSP cost model: global communication

- $P(s,t)$ represents processor (core) t in node s, where $0 \leq s < p_2$ and $0 \leq t < p_1$.
- The cost of a global communication superstep is

$$T_{\text{comm, global}}(h) = hg_2 + l_2,$$

where h is the maximum number of data words sent/received by any node.

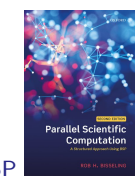


Hybrid-BSP cost model: global computation

- The cost of a global computation superstep is

$$T_{\text{comp, global}} = \max_{0 \le s < p_2} (a_s + b_s g_1 + c_s l_1) + l_2,$$

  where node $P(s, *)$ executes a BSP algorithm with cost $a_s + b_s g_1 + c_s l_1$.
- Taking the maximum of costs instead of summing makes the hybrid-BSP cost model more difficult to use.
- It becomes easier if all nodes perform the same BSP algorithm with the same number of supersteps.

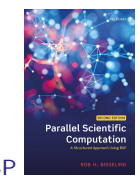


Distribution for hybrid-BSP SpMV

- Every node $P(s,*)$ obtains the components v_j it needs directly from the source processor $P(\phi_{\mathbf{v}}(j))$.
- Component v_j is stored at one designated processor in the node, $P(s,\phi^s_{\mathbf{v}}(j))$, which becomes the local owner of a copy of v_j.
- Every processor that needs v_j obtains it from the local owner in a local communication superstep.
- We define

$$\phi(i,j) = (\phi_0(i,j), \phi_1(i,j))$$

as the owner of $a_{ij} \neq 0$, and $\phi(i,j) = -1$ if $a_{ij} = 0$.



Hybrid-BSP sparse matrix–vector multiplication

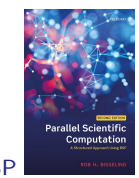
input: A : sparse $n \times n$ matrix, $\mathrm{distr}(A) = \phi = (\phi_0, \phi_1)$,
$\mathbf{v}$: dense vector of length n, $\mathrm{distr}(\mathbf{v}) = \phi_{\mathbf{v}}$.
output: $\mathbf{u}$: dense vector of length n, $\mathbf{u} = A\mathbf{v}$, $\mathrm{distr}(\mathbf{u}) = \phi_{\mathbf{u}}$.

$J_s = \{j : 0 \leq j < n \wedge (\exists i : 0 \leq i < n \wedge \phi_0(i,j) = s)\}$
$0 \leq \phi^s_{\mathbf{v}}(j) < p_1$, for all $j \in J_s$, otherwise $\phi^s_{\mathbf{v}}(j) = -1$

{ Global fanout } ▷ Superstep (0)
for all $j : 0 \leq j < n \wedge \phi^s_{\mathbf{v}}(j) = t$ **do**
    get v_j from $P(\phi_{\mathbf{v}}(j))$;

$J_{st} = \{j : 0 \leq j < n \wedge (\exists i : 0 \leq i < n \wedge \phi(i,j) = (s,t))\}$

{ Local fanout } ▷ Superstep (1)
for all $j \in J_{st}$ **do**
    get v_j from $P(s, \phi^s_{\mathbf{v}}(j))$;



Hybrid-BSP sparse matrix–vector multiplication (cont'd)

$$I_{st} = \{i : 0 \leq i < n \wedge (\exists j : 0 \leq j < n \wedge \phi(i,j) = (s,t))\}$$

{ Local sparse matrix–vector multiplication } ▷ Superstep (2)

for all $i \in I_{st}$ **do**

  $u_{ist} := 0$;

  for all $j : 0 \leq j < n \wedge \phi(i,j) = (s,t)$ **do**

    $u_{ist} := u_{ist} + a_{ij} v_j$;

{ Local fanin } ▷ Superstep (3)

for all $i \in I_{st}$ **do**

  put u_{ist} in $P(s, \phi^s_{\mathbf{u}}(i))$;

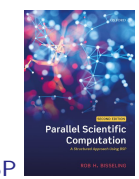
{ Local summation of nonzero partial sums } ▷ Superstep (4)

for all $i : 0 \leq i < n \wedge \phi^s_{\mathbf{u}}(i) = t$ **do**

  $u_{is} := 0$;

  for all $t' : 0 \leq t' < p_1 \wedge u_{ist'} \neq 0$ **do**

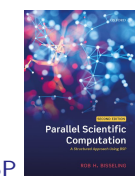
    $u_{is} := u_{is} + u_{ist'}$;



Hybrid-BSP sparse matrix–vector multiplication (cont'd)

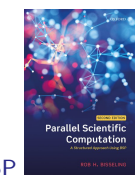
{ Global fanin } ▷ Superstep (5)
for all $i : 0 \leq i < n \wedge \phi^s_{\mathbf{u}}(i) = t$ **do**
    put u_{is} in $P(\phi_{\mathbf{u}}(i))$;

{ Global summation of nonzero partial sums } ▷ Superstep (6)
for all $i : 0 \leq i < n \wedge \phi_{\mathbf{u}}(i) = (s, t)$ **do**
    $u_i := 0$;
    for all $s' : 0 \leq s' < p_2 \wedge u_{is'} \neq 0$ **do**
        $u_i := u_i + u_{is'}$;



Partitioning the data for a hybrid-BSP algorithm

- First, we partition the matrix A using Mondriaan into p_2 parts with an allowed imbalance $\epsilon_2 < \epsilon$, to be discussed later.
- Then, we partition each resulting part s into p_1 parts using Mondriaan with an allowed imbalance $\epsilon_1(s)$ that corresponds to allowing at most $(1+\epsilon)\frac{nz(A)}{p}$ nonzeros per processor.



Communication cost of global and local fanouts

- The communication cost of the global fanout is

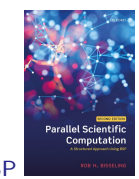
$$T_{(0)} = \frac{V_2 g_2}{p_2},$$

where V_2 is the communication volume, and we assume perfect communication balance.

- The communication cost of the local fanout is

$$T_{(1)} = \frac{V_1 g_1}{p}.$$

Here, V_1 is the total communication volume for all processors in all the nodes, and each processor performs a local h-relation with $h = \frac{V_1}{p_1 p_2} = \frac{V_1}{p}$.



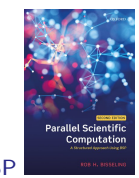
Distributed/shared-memory communication cost ratio β

$$T_{(0)} = \frac{V_2 g_2}{p_2}, \qquad T_{(1)} = \frac{V_1 g_1}{p}.$$

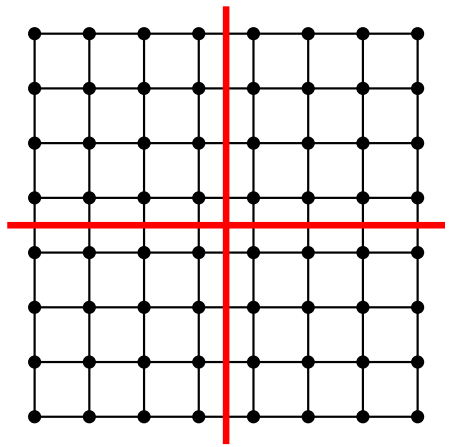
- Communication per data word is more expensive in the distributed-memory global superstep (0) than in the shared-memory local superstep (1), by a factor

$$\beta = \frac{g_2/p_2}{g_1/p} = \frac{g_2 p_1}{g_1}.$$

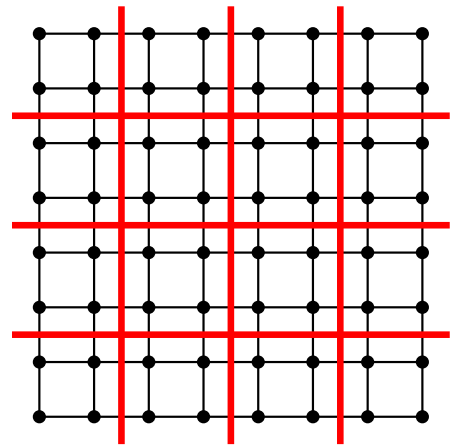
- β depends on p_1, because we model p_1 processors as a single node in the global superstep.



Growth of V as a function of p

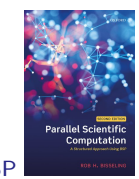


$p = 4$



$p = 16$

- The communication volume grows with p.
- We can model this by a function $V = \mathcal{O}(p^{\alpha})$.
- Unfortunately, the value of α is problem-dependent.
- For 2D Laplacian matrices, $\alpha = 1/2$.
- For 3D, $\alpha = 1/3$



How to choose the initial allowed imbalance ϵ_2?

- Having the equivalent of β times more volume in the global fanout,

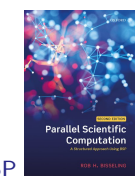
$$V_2 = \beta p_2^\alpha = (\beta^{1/\alpha} p_2)^{1/\alpha},$$

  is the same as having $\hat{q}$ extra levels of recursive bipartitioning, where

$$\hat{q} = \log_2 \beta^{1/\alpha}.$$

- These levels can be considered interspersed with the top $q_2 = \log_2 p_2$ levels.
- Recall that for the first split of a partitioning into $q = \log_2 p$ parts, we choose an allowable imbalance of $\delta = \epsilon/q$.
- Since the total number of levels now increases from $q = \log_2 p$ to $q + \hat{q}$, and we treat all these levels equally, we choose

$$\epsilon_2 = \frac{q_2 + \hat{q}}{q + \hat{q}} \epsilon.$$

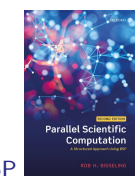


Vector distribution for the fanout

- Our goal is to determine:
  - the unique owner $P(\phi_{\mathbf{v}}(j))$ of the component v_j;
  - for each node $P(s,*)$ with $j \in J_s$, the owner $P(s, \phi^s_{\mathbf{v}}(j))$ of the local copy.
- Matrix partitioning minimizes the communication volume; the subsequent vector partitioning tries to balance the communication load.
- The vector partitioning after the global matrix partitioning determines for each v_j the node $P(s,*)$ that owns it, and we write

$$\phi_{\mathbf{v}}(j) = (s, \phi^s_{\mathbf{v}}(j)).$$

- The vector partitioning after the local matrix partitioning within node $P(s,*)$ then determines $\phi^s_{\mathbf{v}}(j)$.



Summary

- The hybrid-BSP model has a hierarchical architecture with p_2 nodes, each containing p_1 cores.
- A large value of the distributed/shared-memory communication cost ratio $\beta = \frac{g_2 p_1}{g_1}$ is the motivation for the hybrid-BSP model.
- A hybrid-BSP algorithm has local supersteps (with communication and synchronization within a node) and global supersteps.
- For a hybrid-BSP SpMV, this leads to 4 local supersteps and 3 global supersteps.
- The partitioning of the sparse matrix is first done for the nodes, and then for the cores, with a suitable choice of ϵ in each stage.