Towards a highly scalable Multi-BSP fast Fourier transform

Albert-Jan N. Yzelman[1,2], Rob H. Bisseling[3], Dirk Roose[2], and Karl Meerbergen[2]

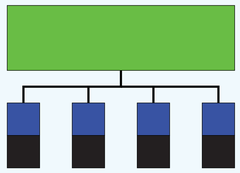
[1] Flanders ExaScience Lab (Intel Labs Europe), Belgium
[2] Department of Computer Science, KU Leuven, Belgium
[3] Department of Mathematics, Utrecht University, the Netherlands

Corresponding author:
Albert-Jan.Yzelman@cs.kuleuven.be

Bulk Synchronous Parallel (BSP)

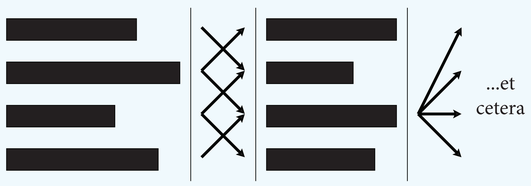
Valiant introduced the BSP model in 1990. The model consists of (1) an abstract **parallel computer**, (2) an imposed structure on **parallel algorithms**, and (3) the resulting **cost function**.

A BSP computer executes p processes concurrently, and does so at a homogeneous speed r. Each process has access to its own local memory. Inter-process communication happens through a full-duplex black-box network.
Preparing this network for all-to-all communication and synchronisation costs l units, while a simultaneous send and receive of one data word costs g units. In illustration, an abstract BSP computer *(p,r,g,l)*:



The network is coloured green, memory is coloured blue, and units that execute the BSP processes are in black.

A BSP algorithm separates local computation phases from communication phases. There is no computation while communicating, and vice versa. In illustration:



The cost of the ith computation phase depends on the BSP process s which was assigned the most work w:

$$T^i_{\text{comp}} = \max_{0 \le s < p} w_i^{(s)}$$

If in the ith communication phase process s transmits t words and receives r words, the ***h*-relation** gives us the linear part of the communication cost in g. Adding the constant latency cost l yields:

$$T^i_{\text{comm}} = \left(\max_{0 \le s < p} \max\{t_i^{(s)}, r_i^{(s)}\} g \right) + l = h_i g + l$$

Fast Fourier transform (FFT)

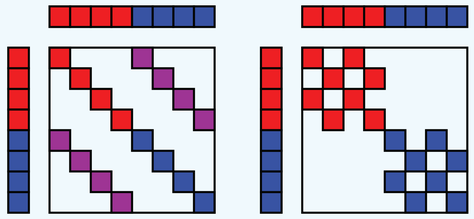
Consider the FFT on complex vectors x, y of power-of-two length n, which computes $y = F_n x$,

$$y_i = \sum_{j=0}^{n-1} x_j e^{-2\pi \imath i j / n}$$

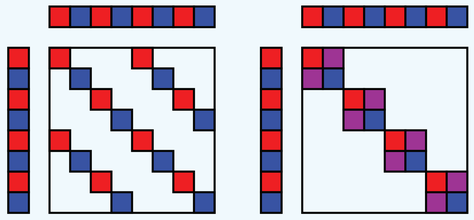
Let the matrix B be the butterfly matrix containing the weights W, and let V be the bit-reversal matrix. Then the FFT can succinctly be written as follows (see also the illustration on the right).

$$F_n x = \left[\Pi_{i=0}^{m-1} \left(I_{2^i} \otimes B_{2^{m-i}} \right) \right] V_n x$$

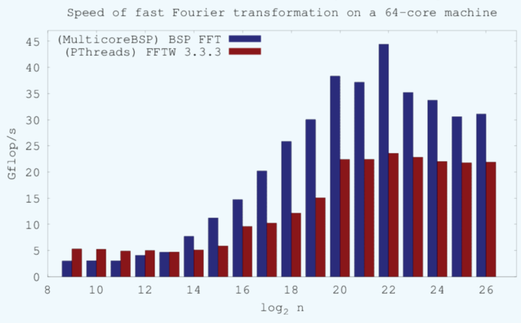
The BSP FFT splits the computation in two phases, and in-between switches from a block distribution to a cyclic one so that communication is minimised:



A block distribution on the last (left) and first stages (right). Purple blocks indicate non-local computations.



Cyclic distribution on the last and first FFT stages



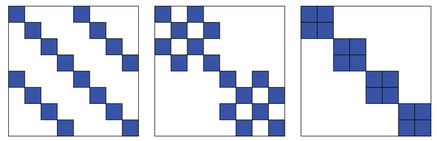
Results on the highly NUMA (8-sockets) HP DL980

$$W_{n/2} = diag(1, e^{-2\pi \imath 1/n}, \ldots, e^{-2\pi \imath (n/2-1)/n})$$

Butterfly matrix weights

$$B_n = \begin{pmatrix} I_{n/2} & W_{n/2} \\ I_{n/2} & -W_{n/2} \end{pmatrix}, \quad S_n x = \begin{pmatrix} x_{\text{even}} \\ x_{\text{odd}} \end{pmatrix}$$

Butterfly matrix (left), even-odd sort matrix (right)



Structure of the butterfly matrices involved for n=8

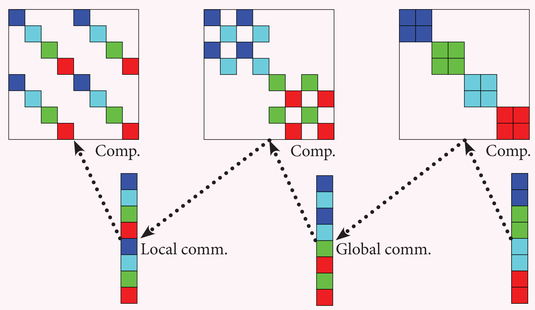
$$V_n = \Pi_{i=1}^{m} \left(I_{2^{m-i}} \otimes S_{2^i} \right)$$

Bit-reversal matrix

$$F_n = \left[\underbrace{\Pi_{i=0}^{t-1} \left(I_{2^i} \otimes B_{2^{m-i}} \right)}_{L_n} \underbrace{\Pi_{i=t}^{m-1} \left(I_{2^i} \otimes B_{2^{m-i}} \right)}_{R_n} \right] V_n$$

For parallelisation, the computation is split in two. For integer m, q *we assume* $n=2^m$ *and* $p=2^q$. *The default value for* t *is* m-q.

The Multi-BSP FFT applies the flat BSP FFT algorithm recursively by splitting the computations L_n and R_n as well. Note only the leaves of the Multi-BSP tree can compute. An illustration for $L=2$, $p_0=2$, $p_1=2$, and $n=8$:



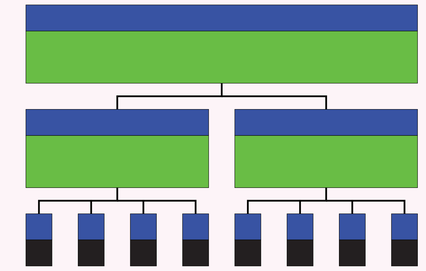
Implementation of the (Multi-BSP) FFT uses **MulticoreBSP for C**, and is freely available at:

www.multicorebsp.com

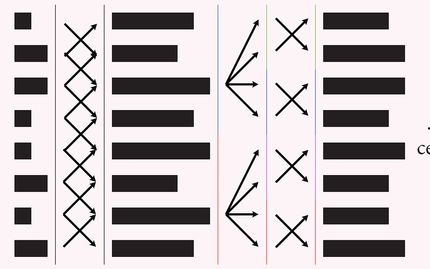
Ref.: A. N. Yzelman, R.H. Bisseling, D. Roose, K. Meerbergen, "MulticoreBSP for C: a high-performance library for shared-memory parallel programming", International Journal of Parallel Programming, in press (2013).

The Multi-BSP model

Modern and future supercomputers consist of nodes with shared-memory parallel capabilities, interconnected by increasingly hierarchical networks.
This gives rise to **Non-Uniform Memory Access** (NUMA) effects on various levels of the compute hierarchy. Valiant updated BSP to account for this situation, resulting in the Multi-BSP model (2008):



An abstract Multi-BSP computer. Each node in the tree is a BSP computer with local memory. Only the leaf nodes contain compute elements.



A Multi-BSP algorithm operating on the various levels of a Multi-BSP computer hierarchy. Successive communication phases are useful when each operates on a different level of the communication network.

Writing L for the number of levels in the Multi-BSP computer hierarchy, N for the number of computation phases, and M_e for the number of communication phases at the eth level of the Multi-BSP tree. Then:

$$T^i_{\text{comm},e} = h_i g_e + l_e$$

This directly leads to the Multi-BSP cost model:

$$T = \sum_{i=0}^{N-1} T^i_{\text{comp}} + \sum_{e=0}^{L-1} \sum_{i=0}^{M_e-1} T^i_{\text{comm},e}$$

If $L=1$ and $M_0=N$, this reduces to the flat BSP model:

$$T = \sum_{i=0}^{N-1} \left(T^i_{\text{comp}} + T^i_{\text{comm}} \right)$$