Introduction to performance modeling

Xavier Álvarez-Farré

Training on energy-aware computing, June 16th, 2023.

High-Performance Computing Team, SURF, Science Park 140, 1098XG Amsterdam, The Netherlands

SURF

Centre Tecnològic de Transferència de Calor
UNIVERSITAT POLITÈCNICA DE CATALUNYA

UNIVERSITY OF TWENTE.

東京工業大学
Tokyo Institute of Technology

Performance modeling

is the process of building analytical or empirical models to predict and quantify the behavior of applications on HPC systems.

- Aims to identify performance bottlenecks and optimize resource utilization.
- Enables understanding of the impact of system configuration and parameters on application performance.
- Facilitates design decisions for system upgrades or new architecture choices.

Performance modeling

is the process of building analytical or empirical models to predict and quantify the behavior of applications on HPC systems.

- Aims to identify performance bottlenecks and optimize resource utilization.
- Enables understanding of the impact of system configuration and parameters on application performance.
- Facilitates design decisions for system upgrades or new architecture choices.

Approaches for performance modeling

- Analytical modeling uses mathematical equations to represent the performance characteristics.
- Empirical modeling uses historical data or benchmarking to build models based on observed behavior.

Performance modeling

is the process of building analytical or empirical models to predict and quantify the behavior of applications on HPC systems.

- Aims to identify performance bottlenecks and optimize resource utilization.
- Enables understanding of the impact of system configuration and parameters on application performance.
- Facilitates design decisions for system upgrades or new architecture choices.

Approaches for performance modeling

- Analytical modeling uses mathematical equations to represent the performance characteristics.
- Empirical modeling uses historical data or benchmarking to build models based on observed behavior.

Challenges in performance modeling

- Heterogeneous architectures and complex interactions between hardware and software components.
- Scalability issues when modeling large-scale systems with millions of concurrent processes.
- Accuracy and validation of models against empirical performance measurements.
- Model portability across different platforms and architectures.

Exploring HPC through vehicle analogies

Sports car

Family van

Transport truck

Sports car

- Power: $400\,\mathrm{hp}$.
- Passengers: $2\,\mathrm{pax}$.

Family van

- Power: $150\,\mathrm{hp}$.
- Passengers: $7\,\mathrm{pax}$.

Transport truck

- Power: $800\,\mathrm{hp}$.
- Passengers: $3\,\mathrm{pax}$.

Exploring HPC through vehicle analogies

Sports car

- Power: $400\,\mathrm{hp}$.
- Passengers: $2\,\mathrm{pax}$.
- Capacity: $0.5\,\mathrm{m}^3$.

Family van

- Power: $150\,\mathrm{hp}$.
- Passengers: $7\,\mathrm{pax}$.
- Capacity: $4\,\mathrm{m}^3$.

Transport truck

- Power: $800\,\mathrm{hp}$.
- Passengers: $3\,\mathrm{pax}$.
- Capacity: $45\,\mathrm{m}^3$.

Exploring HPC through vehicle analogies

Sports car

- Power: $400\,\mathrm{hp}$.
- Passengers: $2\,\mathrm{pax}$.
- Capacity: $0.5\,\mathrm{m}^3$.
- Speed: $350\,\mathrm{m/s}$.

Family van

- Power: $150\,\mathrm{hp}$.
- Passengers: $7\,\mathrm{pax}$.
- Capacity: $4\,\mathrm{m}^3$.
- Speed: $210\,\mathrm{m/s}$.

Transport truck

- Power: $800\,\mathrm{hp}$.
- Passengers: $3\,\mathrm{pax}$.
- Capacity: $45\,\mathrm{m}^3$.
- Speed: $150\,\mathrm{m/2}$.

Exploring HPC through vehicle analogies

Sports car

- Power: 400 hp.

Family van

- Power: 150 hp.

Transport truck

- Power: 800 hp.

Exploring HPC through vehicle analogies

Sports car

- Power: $400\,\mathrm{hp}$.
- Passengers: $2\,\mathrm{pax}$.
- Capacity: $0.5\,\mathrm{m}^3$.
- Speed: $350\,\mathrm{m/s}$.

Family van

- Power: $150\,\mathrm{hp}$.
- Passengers: $7\,\mathrm{pax}$.
- Capacity: $4\,\mathrm{m}^3$.
- Speed: $210\,\mathrm{m/s}$.

Transport truck

- Power: $800\,\mathrm{hp}$.
- Passengers: $3\,\mathrm{pax}$.
- Capacity: $45\,\mathrm{m}^3$.
- Speed: $150\,\mathrm{m/2}$.

Application A

To deliver the maximum power possible.

Exploring HPC through vehicle analogies

Sports car

- Power: $400\,\mathrm{hp}$.
- Passengers: $2\,\mathrm{pax}$.
- Capacity: $0.5\,\mathrm{m}^3$.
- Speed: $350\,\mathrm{m/s}$.

Family van

- Power: $150\,\mathrm{hp}$.
- Passengers: $7\,\mathrm{pax}$.
- Capacity: $4\,\mathrm{m}^3$.
- Speed: $210\,\mathrm{m/s}$.

Transport truck

- Power: $800\,\mathrm{hp}$.
- Passengers: $3\,\mathrm{pax}$.
- Capacity: $45\,\mathrm{m}^3$.
- Speed: $150\,\mathrm{m/2}$.

Application B

To win a speed race on a racing circuit.

Sports car

- Power: $400\,\mathrm{hp}$.
- Passengers: $2\,\mathrm{pax}$.
- Capacity: $0.5\,\mathrm{m}^3$.
- Speed: $350\,\mathrm{m/s}$.

Family van

- Power: $150\,\mathrm{hp}$.
- Passengers: $7\,\mathrm{pax}$.
- Capacity: $4\,\mathrm{m}^3$.
- Speed: $210\,\mathrm{m/s}$.

Transport truck

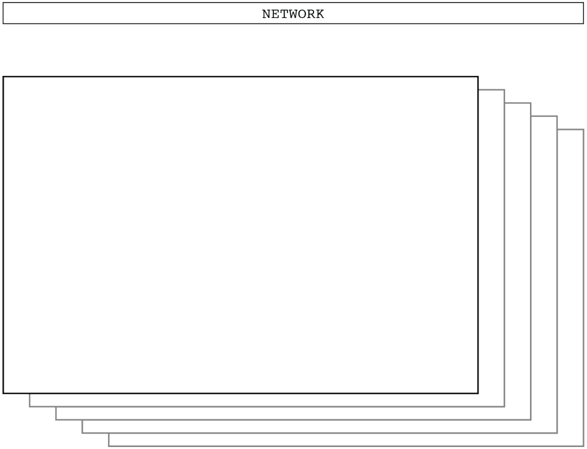
- Power: $800\,\mathrm{hp}$.
- Passengers: $3\,\mathrm{pax}$.
- Capacity: $45\,\mathrm{m}^3$.
- Speed: $150\,\mathrm{m}/2$.

Application C

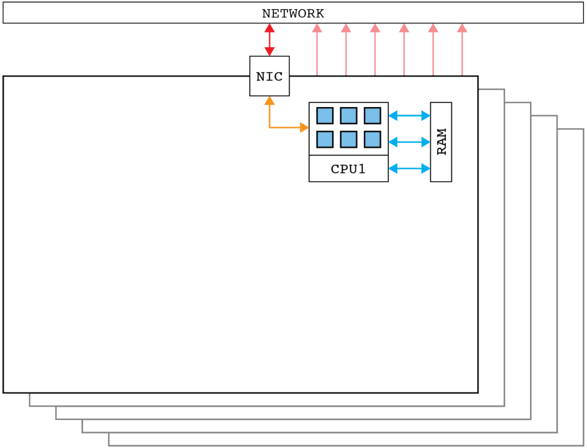
To travel from point A to B with 5 passengers.

Hybrid HPC systems

Modern HPC systems consist of multiple hybrid computing nodes interconnected via a communication infrastructure. The nodes are composed of many hardware devices of different architectures, such as central processing unit (CPU) or graphics processing unit (GPU), among others.

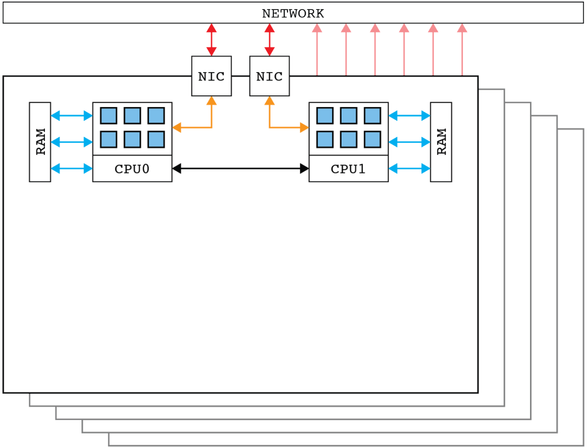


Modern HPC systems consist of multiple hybrid computing nodes interconnected via a communication infrastructure. The nodes are composed of many hardware devices of different architectures, such as central processing unit (CPU) or graphics processing unit (GPU), among others.



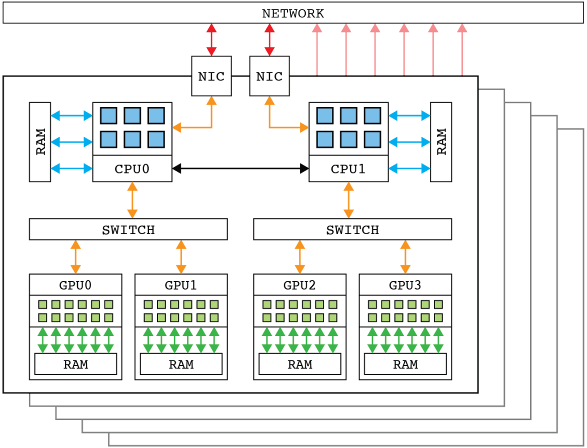
Hybrid HPC systems

Modern HPC systems consist of multiple hybrid computing nodes interconnected via a communication infrastructure. The nodes are composed of many hardware devices of different architectures, such as central processing unit (CPU) or graphics processing unit (GPU), among others.



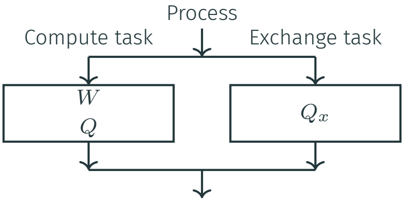
Hybrid HPC systems

Modern HPC systems consist of multiple hybrid computing nodes interconnected via a communication infrastructure. The nodes are composed of many hardware devices of different architectures, such as central processing unit (CPU) or graphics processing unit (GPU), among others.

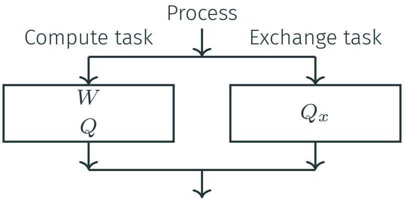


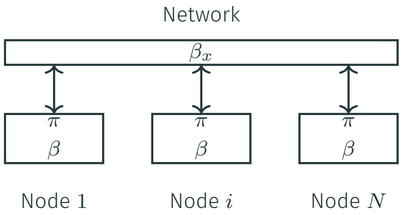
Abstract machine model

Consider computing kernels that perform a vast number of independent operations and memory requests which can be pipelined and parallelized. Distributed-memory parallelization requires large, non-blocking point-to-point messages between subsets of parallel processes that can overlap with bulk computations.



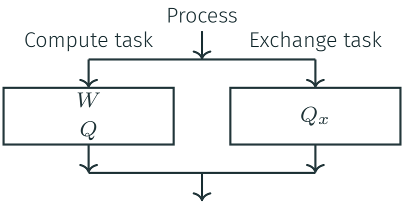
Consider computing kernels that perform a vast number of independent operations and memory requests which can be pipelined and parallelized. Distributed-memory parallelization requires large, non-blocking point-to-point messages between subsets of parallel processes that can overlap with bulk computations.

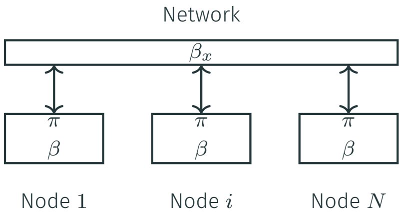




Abstract machine model

Consider computing kernels that perform a vast number of independent operations and memory requests which can be pipelined and parallelized. Distributed-memory parallelization requires large, non-blocking point-to-point messages between subsets of parallel processes that can overlap with bulk computations.

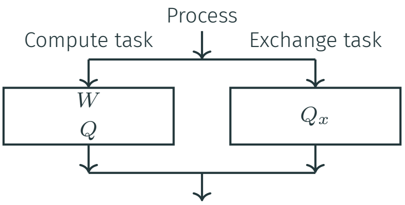


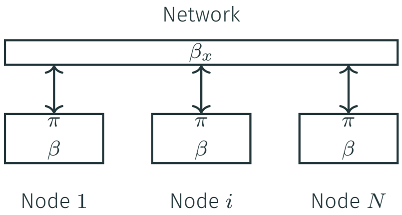


Parameter	Units	Description
W	flop	Number of operations
Q	byte	Main memory traffic
Q_x	byte	Network traffic

Abstract machine model

Consider computing kernels that perform a vast number of independent operations and memory requests which can be pipelined and parallelized. Distributed-memory parallelization requires large, non-blocking point-to-point messages between subsets of parallel processes that can overlap with bulk computations.





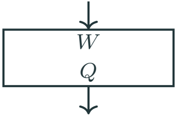
Parameter	Units	Description
W	flop	Number of operations
Q	byte	Main memory traffic
Q_x	byte	Network traffic

Parameter	Units	Description
π	flop/s	Peak performance
β	byte/s	Main memory bandwidth
β_x	byte/s	Network bandwidth

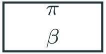
Abstract machine model

Consider computing kernels that perform a vast number of independent operations and memory requests which can be pipelined and parallelized.

Compute task



Node

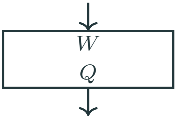


Parameter	Units	Description
W	flop	Number of operations
Q	byte	Main memory traffic

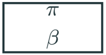
Abstract machine model

Consider computing kernels that perform a vast number of independent operations and memory requests which can be pipelined and parallelized.

Compute task



Node



Parameter	Units	Description
W	flop	Number of operations
Q	byte	Main memory traffic

Parameter	Units	Description
π	flop/s	Peak performance
β	byte/s	Main memory bandwidth

axpy

The axpy kernel performing the linear combination of two vectors reads:

$$\boldsymbol{y} \leftarrow \alpha \boldsymbol{x} + \boldsymbol{y},$$

The axpy kernel performing the linear combination of two vectors reads:

$$\boldsymbol{y} \leftarrow \alpha \boldsymbol{x} + \boldsymbol{y},$$

where α is a constant, and $\boldsymbol{x}$ and $\boldsymbol{y}$ are two vectors of size n. In such an operation, the processor performs:

- Work (W): n additions and n multiplications, that is, $2n$ floating-point operations.
- Memory traffic (Q): $2n$ elements read and n written, that is, $3n \cdot 8\,\mathrm{bytes}$ in double precision.

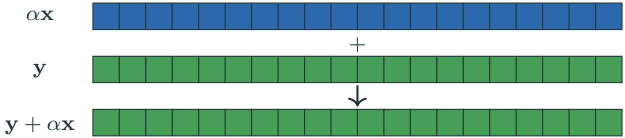
The axpy kernel performing the linear combination of two vectors reads:

$$\boldsymbol{y} \leftarrow \alpha \boldsymbol{x} + \boldsymbol{y},$$

where α is a constant, and $\boldsymbol{x}$ and $\boldsymbol{y}$ are two vectors of size n. In such an operation, the processor performs:

- Work (W): n additions and n multiplications, that is, $2n$ floating-point operations.
- Memory traffic (Q): $2n$ elements read and n written, that is, $3n \cdot 8\,\mathrm{bytes}$ in double precision.

The kernel is illustrated as follows.



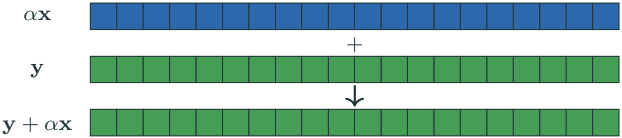
The axpy kernel performing the linear combination of two vectors reads:

$$\boldsymbol{y} \leftarrow \alpha \boldsymbol{x} + \boldsymbol{y},$$

where α is a constant, and $\boldsymbol{x}$ and $\boldsymbol{y}$ are two vectors of size n. In such an operation, the processor performs:

- Work (W): n additions and n multiplications, that is, $2n$ floating-point operations.
- Memory traffic (Q): $2n$ elements read and n written, that is, $3n \cdot 8\,\text{bytes}$ in double precision.

The kernel is illustrated as follows.



The time required to perform the calculations is:

$$t^W = \frac{W}{\pi} = \frac{2n}{\pi}.$$

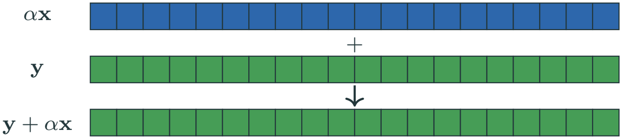
The axpy kernel performing the linear combination of two vectors reads:

$$\boldsymbol{y} \leftarrow \alpha \boldsymbol{x} + \boldsymbol{y},$$

where α is a constant, and $\boldsymbol{x}$ and $\boldsymbol{y}$ are two vectors of size n. In such an operation, the processor performs:

- Work (W): n additions and n multiplications, that is, $2n$ floating-point operations.
- Memory traffic (Q): $2n$ elements read and n written, that is, $3n \cdot 8\,\mathrm{bytes}$ in double precision.

The kernel is illustrated as follows.



The time required to perform the calculations is:

$$t^W = \frac{W}{\pi} = \frac{2n}{\pi}.$$

The time required to read and write the data is:

$$t^Q = \frac{Q}{\beta} = \frac{3n \cdot 8}{\beta}.$$

axpy

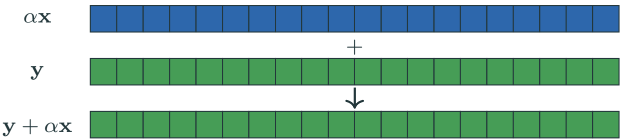
The axpy kernel performing the linear combination of two vectors reads:

$$\boldsymbol{y} \leftarrow \alpha \boldsymbol{x} + \boldsymbol{y},$$

where α is a constant, and $\boldsymbol{x}$ and $\boldsymbol{y}$ are two vectors of size n. In such an operation, the processor performs:

- Work (W): n additions and n multiplications, that is, $2n$ floating-point operations.
- Memory traffic (Q): $2n$ elements read and n written, that is, $3n \cdot 8\,\text{bytes}$ in double precision.

The kernel is illustrated as follows.



The time required to perform the calculations is:

$$t^W = \frac{W}{\pi} = \frac{2n}{\pi}.$$

The time required to read and write the data is:

$$t^Q = \frac{Q}{\beta} = \frac{3n \cdot 8}{\beta}.$$

The time required to execute the kernel:

$$t = max(t^W, t^Q) = max(\frac{2n}{\pi}, \frac{3n \cdot 8}{\beta}).$$

The gemm kernel performing the general matrix multiplication reads:

$$\mathsf{C} \leftarrow \alpha \mathsf{AB} + \beta \mathsf{C},$$

The gemm kernel performing the general matrix multiplication reads:

$$\mathsf{C} \leftarrow \alpha \mathsf{AB} + \beta \mathsf{C},$$

where α and β are constants, and $\mathsf{A} \in \mathbb{R}^{m \times k}$, $\mathsf{B} \in \mathbb{R}^{k \times n}$, and $\mathsf{C} \in \mathbb{R}^{m \times n}$ are matrices. In such an operation, the processor performs:

- Work (W): $mn(k+3)$ additions and multiplications, that is, approximately, $2mnk$ floating-point operations.
- Memory traffic (Q): $mk + kn + mn$ elements read and mn written, that is, $(mk + kn + 2mn) \cdot 8\,\mathrm{bytes}$ in double precision.

gemm

The gemm kernel performing the general matrix multiplication reads:

$$\mathsf{C} \leftarrow \alpha \mathsf{AB} + \beta \mathsf{C},$$

where α and β are constants, and $\mathsf{A} \in \mathbb{R}^{m \times k}$, $\mathsf{B} \in \mathbb{R}^{k \times n}$, and $\mathsf{C} \in \mathbb{R}^{m \times n}$ are matrices. In such an operation, the processor performs:

- Work (W): $mn(k+3)$ additions and multiplications, that is, approximately, $2mnk$ floating-point operations.
- Memory traffic (Q): $mk + kn + mn$ elements read and mn written, that is, $(mk + kn + 2mn) \cdot 8\,\mathrm{bytes}$ in double precision.

The kernel is illustrated as follows.

$$\begin{array}{|c|c|}\hline \mathsf{A}_{1,1} & \mathsf{A}_{1,2} \\ \hline \mathsf{A}_{2,1} & \mathsf{A}_{2,2} \\ \hline \mathsf{A}_{3,1} & \mathsf{A}_{3,2} \\ \hline \end{array} \times \begin{array}{|c|c|c|}\hline \mathsf{B}_{1,1} & \mathsf{B}_{1,2} & \mathsf{B}_{1,3} \\ \hline \mathsf{B}_{2,1} & \mathsf{B}_{2,2} & \mathsf{B}_{2,3} \\ \hline \end{array} + \begin{array}{|c|c|c|}\hline \mathsf{C}_{1,1} & \mathsf{C}_{1,2} & \mathsf{C}_{1,3} \\ \hline \mathsf{C}_{2,1} & \mathsf{C}_{2,2} & \mathsf{C}_{2,3} \\ \hline \mathsf{C}_{3,1} & \mathsf{C}_{3,2} & \mathsf{C}_{3,3} \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|}\hline \mathsf{C}_{1,1} & \mathsf{C}_{1,2} & \mathsf{C}_{1,3} \\ \hline \mathsf{C}_{2,1} & \mathsf{C}_{2,2} & \mathsf{C}_{2,3} \\ \hline \mathsf{C}_{3,1} & \mathsf{C}_{3,2} & \mathsf{C}_{3,3} \\ \hline \end{array}$$

gemm

The gemm kernel performing the general matrix multiplication reads:

$$\mathsf{C} \leftarrow \alpha \mathsf{A}\mathsf{B} + \beta \mathsf{C},$$

where α and β are constants, and $\mathsf{A} \in \mathbb{R}^{m\times k}$, $\mathsf{B} \in \mathbb{R}^{k\times n}$, and $\mathsf{C} \in \mathbb{R}^{m\times n}$ are matrices. In such an operation, the processor performs:

- Work (W): $mn(k+3)$ additions and multiplications, that is, approximately, $2mnk$ floating-point operations.
- Memory traffic (Q): $mk + kn + mn$ elements read and mn written, that is, $(mk + kn + 2mn) \cdot 8\,\mathrm{bytes}$ in double precision.

The kernel is illustrated as follows.

$$\begin{bmatrix} \mathsf{A}_{1,1} & \mathsf{A}_{1,2} \\ \mathsf{A}_{2,1} & \mathsf{A}_{2,2} \\ \mathsf{A}_{3,1} & \mathsf{A}_{3,2} \end{bmatrix} \times \begin{bmatrix} \mathsf{B}_{1,1} & \mathsf{B}_{1,2} & \mathsf{B}_{1,3} \\ \mathsf{B}_{2,1} & \mathsf{B}_{2,2} & \mathsf{B}_{2,3} \end{bmatrix} + \begin{bmatrix} \mathsf{C}_{1,1} & \mathsf{C}_{1,2} & \mathsf{C}_{1,3} \\ \mathsf{C}_{2,1} & \mathsf{C}_{2,2} & \mathsf{C}_{2,3} \\ \mathsf{C}_{3,1} & \mathsf{C}_{3,2} & \mathsf{C}_{3,3} \end{bmatrix} \rightarrow \begin{bmatrix} \mathsf{C}_{1,1} & \mathsf{C}_{1,2} & \mathsf{C}_{1,3} \\ \mathsf{C}_{2,1} & \mathsf{C}_{2,2} & \mathsf{C}_{2,3} \\ \mathsf{C}_{3,1} & \mathsf{C}_{3,2} & \mathsf{C}_{3,3} \end{bmatrix}$$

The time required to perform the calculations is:

$$t^W = \frac{W}{\pi} = \frac{2mnk}{\pi}.$$

gemm

The gemm kernel performing the general matrix multiplication reads:

$$\mathsf{C} \leftarrow \alpha \mathsf{AB} + \beta \mathsf{C},$$

where α and β are constants, and $\mathsf{A} \in \mathbb{R}^{m \times k}$, $\mathsf{B} \in \mathbb{R}^{k \times n}$, and $\mathsf{C} \in \mathbb{R}^{m \times n}$ are matrices. In such an operation, the processor performs:

- Work (W): $mn(k+3)$ additions and multiplications, that is, approximately, $2mnk$ floating-point operations.
- Memory traffic (Q): $mk + kn + mn$ elements read and mn written, that is, $(mk + kn + 2mn) \cdot 8\,\text{bytes}$ in double precision.

The kernel is illustrated as follows.

$$\begin{bmatrix} \mathsf{A}_{1,1} & \mathsf{A}_{1,2} \\ \mathsf{A}_{2,1} & \mathsf{A}_{2,2} \\ \mathsf{A}_{3,1} & \mathsf{A}_{3,2} \end{bmatrix} \times \begin{bmatrix} \mathsf{B}_{1,1} & \mathsf{B}_{1,2} & \mathsf{B}_{1,3} \\ \mathsf{B}_{2,1} & \mathsf{B}_{2,2} & \mathsf{B}_{2,3} \end{bmatrix} + \begin{bmatrix} \mathsf{C}_{1,1} & \mathsf{C}_{1,2} & \mathsf{C}_{1,3} \\ \mathsf{C}_{2,1} & \mathsf{C}_{2,2} & \mathsf{C}_{2,3} \\ \mathsf{C}_{3,1} & \mathsf{C}_{3,2} & \mathsf{C}_{3,3} \end{bmatrix} \rightarrow \begin{bmatrix} \mathsf{C}_{1,1} & \mathsf{C}_{1,2} & \mathsf{C}_{1,3} \\ \mathsf{C}_{2,1} & \mathsf{C}_{2,2} & \mathsf{C}_{2,3} \\ \mathsf{C}_{3,1} & \mathsf{C}_{3,2} & \mathsf{C}_{3,3} \end{bmatrix}$$

The time required to perform the calculations is:

$$t^W = \frac{W}{\pi} = \frac{2mnk}{\pi}.$$

The time required to read and write the data is:

$$t^Q = \frac{Q}{\beta} = \frac{8(mk + kn + 2mn)}{\beta}.$$

gemm

The gemm kernel performing the general matrix multiplication reads:

$$\mathsf{C} \leftarrow \alpha \mathsf{AB} + \beta \mathsf{C},$$

where α and β are constants, and $\mathsf{A} \in \mathbb{R}^{m \times k}$, $\mathsf{B} \in \mathbb{R}^{k \times n}$, and $\mathsf{C} \in \mathbb{R}^{m \times n}$ are matrices. In such an operation, the processor performs:

- Work (W): $mn(k+3)$ additions and multiplications, that is, approximately, $2mnk$ floating-point operations.
- Memory traffic (Q): $mk + kn + mn$ elements read and mn written, that is, $(mk + kn + 2mn) \cdot 8\,\text{bytes}$ in double precision.

The kernel is illustrated as follows.

$$\begin{bmatrix} \mathsf{A}_{1,1} & \mathsf{A}_{1,2} \\ \mathsf{A}_{2,1} & \mathsf{A}_{2,2} \\ \mathsf{A}_{3,1} & \mathsf{A}_{3,2} \end{bmatrix} \times \begin{bmatrix} \mathsf{B}_{1,1} & \mathsf{B}_{1,2} & \mathsf{B}_{1,3} \\ \mathsf{B}_{2,1} & \mathsf{B}_{2,2} & \mathsf{B}_{2,3} \end{bmatrix} + \begin{bmatrix} \mathsf{C}_{1,1} & \mathsf{C}_{1,2} & \mathsf{C}_{1,3} \\ \mathsf{C}_{2,1} & \mathsf{C}_{2,2} & \mathsf{C}_{2,3} \\ \mathsf{C}_{3,1} & \mathsf{C}_{3,2} & \mathsf{C}_{3,3} \end{bmatrix} \rightarrow \begin{bmatrix} \mathsf{C}_{1,1} & \mathsf{C}_{1,2} & \mathsf{C}_{1,3} \\ \mathsf{C}_{2,1} & \mathsf{C}_{2,2} & \mathsf{C}_{2,3} \\ \mathsf{C}_{3,1} & \mathsf{C}_{3,2} & \mathsf{C}_{3,3} \end{bmatrix}$$

The time required to perform the calculations is:

$$t^W = \frac{W}{\pi} = \frac{2mnk}{\pi}.$$

The time required to read and write the data is:

$$t^Q = \frac{Q}{\beta} = \frac{8(mk + kn + 2mn)}{\beta}.$$

The time required to execute the kernel:

$$t = max(\frac{2mnk}{\pi}, \frac{8(mk + kn + 2mn)}{\beta}).$$

The roofline model

The elapsed time of a kernel running on a single computing unit is estimated as follows:

$$t^{RL} = max\left(\frac{W}{\pi}, \frac{Q}{\beta}\right),$$

The elapsed time of a kernel running on a single computing unit is estimated as follows:

$$t^{RL} = max\left(\frac{W}{\pi}, \frac{Q}{\beta}\right),$$

Application performance is often considered more relevant than elapsed time itself. Let's divide the work W by t^{RL}:

$$R^{RL} = \frac{W}{t^{RL}} = min\left(\pi, \frac{W}{Q}\beta\right) = min(\pi, I\beta).$$

The model introduces operational intensity, I, as the ratio of floating-point operations, in flops, to the memory traffic, in bytes.

The elapsed time of a kernel running on a single computing unit is estimated as follows:

$$t^{RL} = max\left(\frac{W}{\pi}, \frac{Q}{\beta}\right),$$

Application performance is often considered more relevant than elapsed time itself. Let's divide the work W by t^{RL}:

$$R^{RL} = \frac{W}{t^{RL}} = min\left(\pi, \frac{W}{Q}\beta\right) = min(\pi, I\beta).$$

The model introduces operational intensity, I, as the ratio of floating-point operations, in flops, to the memory traffic, in bytes.

Operational intensity

Fundamental characteristic of the algorithm and determines its theoretically achievable performance.

The elapsed time of a kernel running on a single computing unit is estimated as follows:

$$t^{RL} = max\left(\frac{W}{\pi}, \frac{Q}{\beta}\right),$$

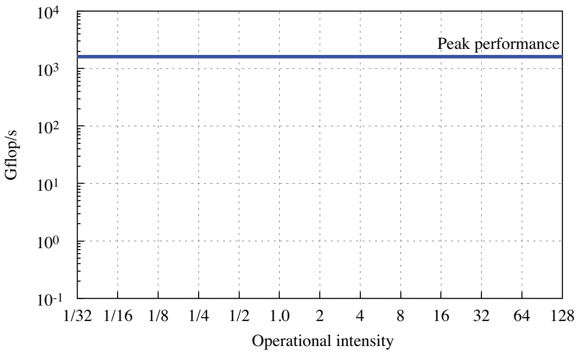
Application performance is often considered more relevant than elapsed time itself. Let's divide the work W by t^{RL}:

$$R^{RL} = \frac{W}{t^{RL}} = min\left(\pi, \frac{W}{Q}\beta\right) = min(\pi, I\beta).$$

The model introduces operational intensity, I, as the ratio of floating-point operations, in flops, to the memory traffic, in bytes.

Operational intensity

Fundamental characteristic of the algorithm and determines its theoretically achievable performance.



The elapsed time of a kernel running on a single computing unit is estimated as follows:

$$t^{RL} = max\left(\frac{W}{\pi}, \frac{Q}{\beta}\right),$$

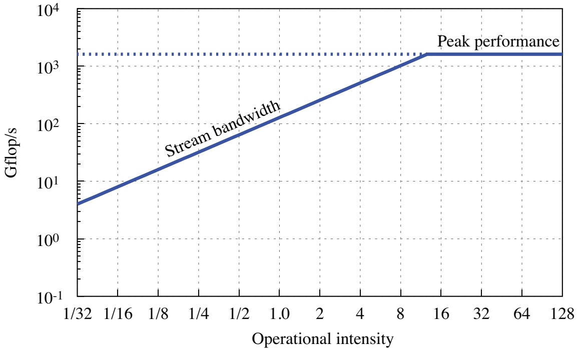
Application performance is often considered more relevant than elapsed time itself. Let's divide the work W by t^{RL}:

$$R^{RL} = \frac{W}{t^{RL}} = min\left(\pi, \frac{W}{Q}\beta\right) = min(\pi, I\beta).$$

The model introduces operational intensity, I, as the ratio of floating-point operations, in flops, to the memory traffic, in bytes.

Operational intensity

Fundamental characteristic of the algorithm and determines its theoretically achievable performance.



The elapsed time of a kernel running on a single computing unit is estimated as follows:

$$t^{RL} = max\left(\frac{W}{\pi}, \frac{Q}{\beta}\right),$$

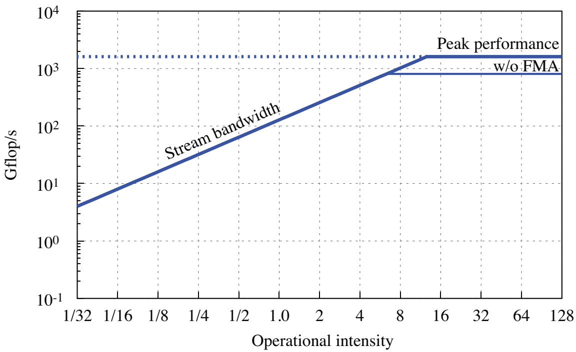
Application performance is often considered more relevant than elapsed time itself. Let's divide the work W by t^{RL}:

$$R^{RL} = \frac{W}{t^{RL}} = min\left(\pi, \frac{W}{Q}\beta\right) = min(\pi, I\beta).$$

The model introduces operational intensity, I, as the ratio of floating-point operations, in flops, to the memory traffic, in bytes.

Operational intensity

Fundamental characteristic of the algorithm and determines its theoretically achievable performance.



The elapsed time of a kernel running on a single computing unit is estimated as follows:

$$t^{RL} = max\left(\frac{W}{\pi}, \frac{Q}{\beta}\right),$$

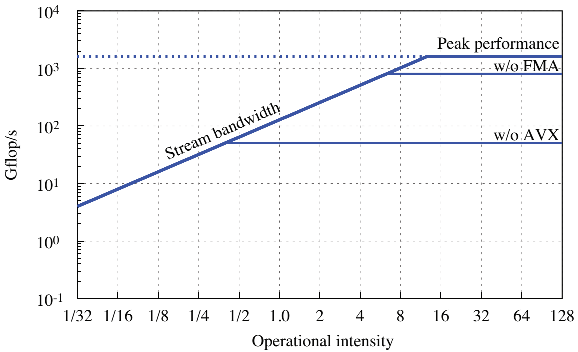
Application performance is often considered more relevant than elapsed time itself. Let's divide the work W by t^{RL}:

$$R^{RL} = \frac{W}{t^{RL}} = min\left(\pi, \frac{W}{Q}\beta\right) = min(\pi, I\beta).$$

The model introduces operational intensity, I, as the ratio of floating-point operations, in flops, to the memory traffic, in bytes.

Operational intensity

Fundamental characteristic of the algorithm and determines its theoretically achievable performance.



The elapsed time of a kernel running on a single computing unit is estimated as follows:

$$t^{RL} = max\left(\frac{W}{\pi}, \frac{Q}{\beta}\right),$$

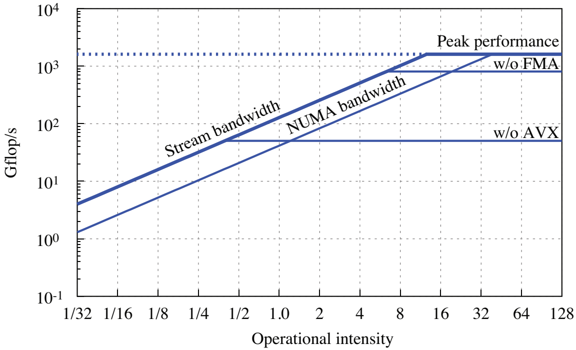
Application performance is often considered more relevant than elapsed time itself. Let's divide the work W by t^{RL}:

$$R^{RL} = \frac{W}{t^{RL}} = min\left(\pi, \frac{W}{Q}\beta\right) = min(\pi, I\beta).$$

The model introduces operational intensity, I, as the ratio of floating-point operations, in flops, to the memory traffic, in bytes.

Operational intensity

Fundamental characteristic of the algorithm and determines its theoretically achievable performance.



The elapsed time of a kernel running on a single computing unit is estimated as follows:

$$t^{RL} = max\left(\frac{W}{\pi}, \frac{Q}{\beta}\right),$$

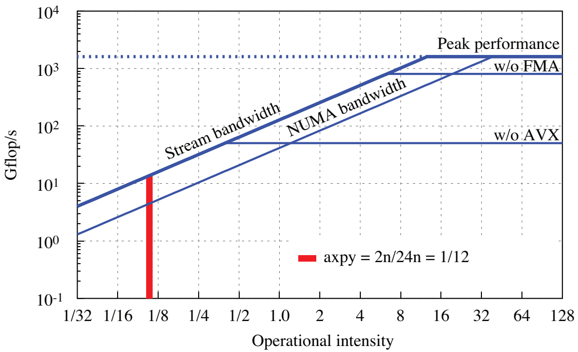
Application performance is often considered more relevant than elapsed time itself. Let's divide the work W by t^{RL}:

$$R^{RL} = \frac{W}{t^{RL}} = min\left(\pi, \frac{W}{Q}\beta\right) = min(\pi, I\beta).$$

The model introduces operational intensity, I, as the ratio of floating-point operations, in flops, to the memory traffic, in bytes.

Operational intensity

Fundamental characteristic of the algorithm and determines its theoretically achievable performance.



The elapsed time of a kernel running on a single computing unit is estimated as follows:

$$t^{RL} = max\left(\frac{W}{\pi}, \frac{Q}{\beta}\right),$$

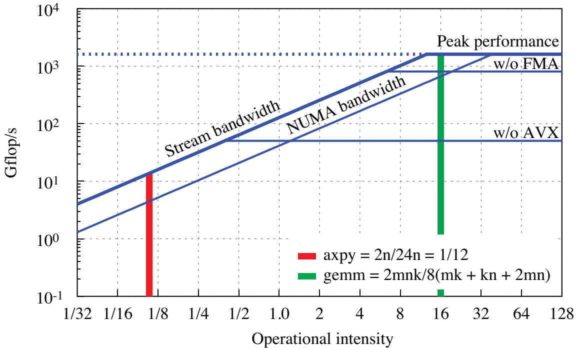
Application performance is often considered more relevant than elapsed time itself. Let's divide the work W by t^{RL}:

$$R^{RL} = \frac{W}{t^{RL}} = min\left(\pi, \frac{W}{Q}\beta\right) = min(\pi, I\beta).$$

The model introduces operational intensity, I, as the ratio of floating-point operations, in flops, to the memory traffic, in bytes.

Operational intensity

Fundamental characteristic of the algorithm and determines its theoretically achievable performance.



The elapsed time of a kernel running on a single computing unit is estimated as follows:

$$t^{RL} = max\left(\frac{W}{\pi}, \frac{Q}{\beta}\right),$$

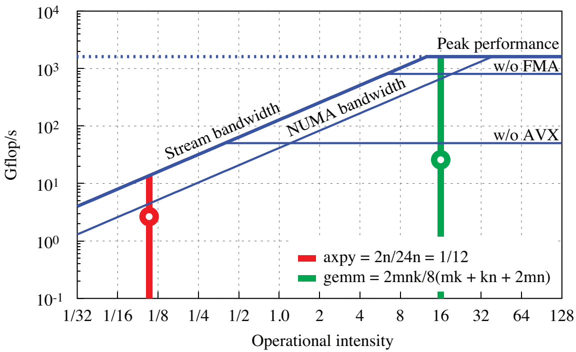
Application performance is often considered more relevant than elapsed time itself. Let's divide the work W by t^{RL}:

$$R^{RL} = \frac{W}{t^{RL}} = min\left(\pi, \frac{W}{Q}\beta\right) = min(\pi, I\beta).$$

The model introduces operational intensity, I, as the ratio of floating-point operations, in flops, to the memory traffic, in bytes.

Operational intensity

Fundamental characteristic of the algorithm and determines its theoretically achievable performance.



The elapsed time of a kernel running on a single computing unit is estimated as follows:

$$t^{RL} = max\left(\frac{W}{\pi}, \frac{Q}{\beta}\right),$$

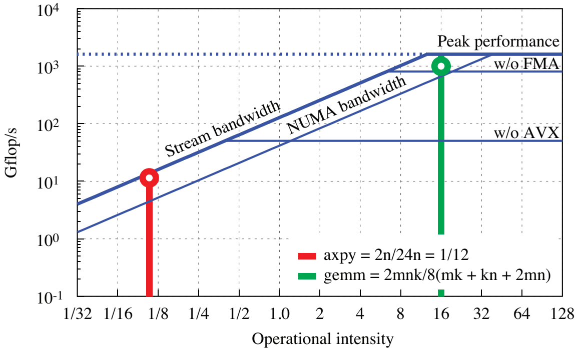
Application performance is often considered more relevant than elapsed time itself. Let's divide the work W by t^{RL}:

$$R^{RL} = \frac{W}{t^{RL}} = min\left(\pi, \frac{W}{Q}\beta\right) = min(\pi, I\beta).$$

The model introduces operational intensity, I, as the ratio of floating-point operations, in flops, to the memory traffic, in bytes.

Operational intensity

Fundamental characteristic of the algorithm and determines its theoretically achievable performance.



HPCG benchmark, November 2022

Rank	Site	Computer	Cores	HPL Rmax (Pflop/s)	TOP500 Rank	HPCG (Pflop/s)	Fraction of Peak
1	RIKEN Center for Computational Science **Japan**	**Supercomputer Fugaku** — A64FX 48C 2.2GHz, Tofu interconnect D	7,630,848	442.01	2	16.00	3.0%
2	DOE/SC/Oak Ridge National Laboratory **United States**	**Frontier** — AMD Optimized 3rd Generation EPYC 64C 2GHz, Slingshot-11, AMD Instinct MI250X	8,730,112	1102.00	1	14.05	0.8%
3	EuroHPC/CSC **Finland**	**LUMI** — AMD Optimized 3rd Generation EPYC 64C 2GHz, Slingshot-11, AMD Instinct MI250X	2,220,288	309.10	3	3.408	0.8%
4	DOE/SC/Oak Ridge National Laboratory **United States**	**Summit** — IBM POWER9 22C 3.07GHz, Dual-rail Mellanox EDR Infiniband, NVIDIA Volta GV100	2,414,592	148.60	5	2.926	1.5%
5	EuroHPC/CINECA **Italy**	**Leonardo** — Xeon Platinum 8358 32C 2.6GHz, Quad-rail NVIDIA HDR100 Infiniband, NVIDIA A100 SXM4 64 GB	1,463,616	174.70	4	2.567	1.0%

Performance modeling

is the process of building analytical or empirical models to predict and quantify the behavior of applications on HPC systems.

- Hardware architectures and system configurations are increasingly complex. Performance modeling aims to develop comprehensive and simplified machine models.

SURF

Performance modeling

is the process of building analytical or empirical models to predict and quantify the behavior of applications on HPC systems.

- Hardware architectures and system configurations are increasingly complex. Performance modeling aims to develop comprehensive and simplified machine models.
- Energy and power constraints require understanding and optimizing application's performance.

Performance modeling

is the process of building analytical or empirical models to predict and quantify the behavior of applications on HPC systems.

- Hardware architectures and system configurations are increasingly complex. Performance modeling aims to develop comprehensive and simplified machine models.
- Energy and power constraints require understanding and optimizing application's performance.
- Performance modeling provides insights into application behavior on HPC systems.

Performance modeling

is the process of building analytical or empirical models to predict and quantify the behavior of applications on HPC systems.

- Hardware architectures and system configurations are increasingly complex. Performance modeling aims to develop comprehensive and simplified machine models.
- Energy and power constraints require understanding and optimizing application's performance.
- Performance modeling provides insights into application behavior on HPC systems.
- Understand your application's nature and requirements for accurate performance modeling.

Performance modeling

is the process of building analytical or empirical models to predict and quantify the behavior of applications on HPC systems.

- Hardware architectures and system configurations are increasingly complex. Performance modeling aims to develop comprehensive and simplified machine models.
- Energy and power constraints require understanding and optimizing application's performance.
- Performance modeling provides insights into application behavior on HPC systems.
- Understand your application's nature and requirements for accurate performance modeling.
- Choose specific models for each application based on its characteristics.

Performance modeling

is the process of building analytical or empirical models to predict and quantify the behavior of applications on HPC systems.

- Hardware architectures and system configurations are increasingly complex. Performance modeling aims to develop comprehensive and simplified machine models.
- Energy and power constraints require understanding and optimizing application's performance.
- Performance modeling provides insights into application behavior on HPC systems.
- Understand your application's nature and requirements for accurate performance modeling.
- Choose specific models for each application based on its characteristics.
- Analytical models, like the roofline model, identify performance bottlenecks and guide optimization efforts.

Performance modeling

is the process of building analytical or empirical models to predict and quantify the behavior of applications on HPC systems.

- Hardware architectures and system configurations are increasingly complex. Performance modeling aims to develop comprehensive and simplified machine models.
- Energy and power constraints require understanding and optimizing application's performance.
- Performance modeling provides insights into application behavior on HPC systems.
- Understand your application's nature and requirements for accurate performance modeling.
- Choose specific models for each application based on its characteristics.
- Analytical models, like the roofline model, identify performance bottlenecks and guide optimization efforts.
- Identify parallelism, optimize resource utilization, and improve system efficiency through performance modeling.

Thank you for your attention