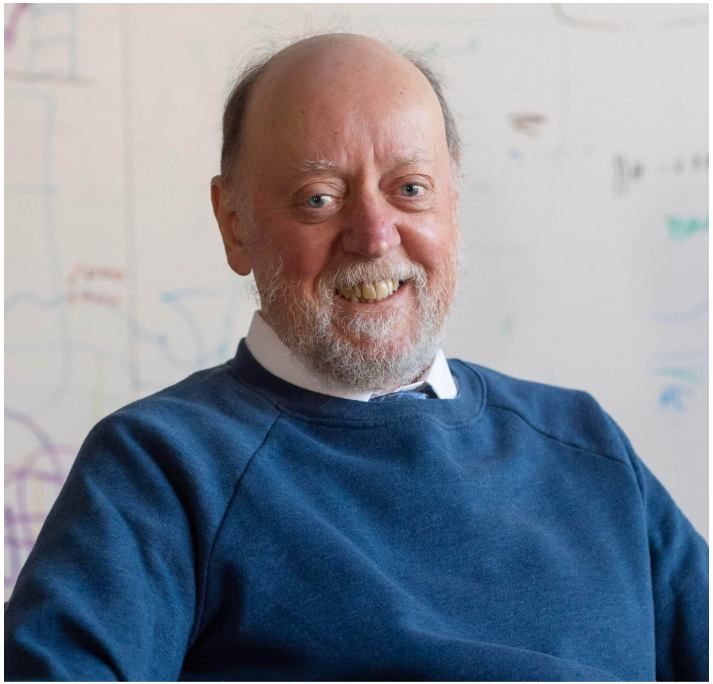


J. J. Dongarra

ACM Turing Award Recipient, 2021
A Tribute by Alvaro Coutinho

If you don't measure it, you don't know it[1]

[1]My version of Peter Drucker's: If you can't measure it you can't improve it

Contents

- Who is Jack Dongarra?
- What Dongarra has done to earn the ACM Turing Award?
- Why this is important?
- Discussion

Who is Jack Dongarra[1]

- Born in 1950 in Chicago, IL, USA, the oldest of three boys. All of his grandparents were originally from Sicily, Italy. His father arrived in America as a teenager. He attended a Catholic elementary school, Saint Daniel the Prophet, for eight years. Most probably due to undiagnosed dyslexia, he was not a good student, particularly in skills such as writing and reading.
- He went to a public high school, Kennedy High School, southwest Chicago, where he discovered science! After high school he decided to become a high school teacher. He then went to a Chicago college that produced many of the teachers in the Chicago public school system: Chicago State University.

[1]Dongarra's interview, SIAM, 2004, donated to the Archives of Computer History: http://archive.computerhistory.org/resources/access/text/2013/12/102746788-05-01-acc.pdf

Who is Jack Dongarra: the early years

- At Chicago State he took science and math courses and decided to go on and get a Master's. However, his physics professor suggested him to apply for a 16-week undergrad position at the Appl Math Division at Argonne National Lab.
- There he met Brian Smith, a scientist at the lab, and the unofficial head of the EISPACK project. EISPACK is a collection of mathematical software that was developed based on algorithms and ideas that had been through the community mainly developed by Wilkinson and Reinsch, in the book "The Handbook for Automatic Computation." All algorithms in this book were in ALGOL and the project translated them in FORTRAN.
- Eventually he got a bachelor's degree in Mathematics from Chicago State in 1972.

Who is Jack Dongarra: discovering computer science

- With that experience in Argonne, he decided to get a degree in Computer Science and applied to the Illinois Institute of Technology and Chicago University.
- He was accepted in both, but went to IIT, where he could get some financial support. Also, during his undergrad years, he worked to support himself.
- At IIT, he took courses and really got immersed in computer science, and some of the more theoretical aspects of the field. He tried to learn as much as he could about numerical methods, about computing in general, and about the hardware, software, and theory aspects of computing.
- He got his Master’s degree in Computer Science from IIT in 1973.

Who is Jack Dongarra: got his PhD, ready to shine

- His masters' project was on an out-of-core algorithm for reducing a large banded matrix to tridiagonal form in preparation for finding the eigenvalues of the matrix.
- After the Master's he was invited to join Argonne, working on EISPACK, on an IBM 360/75.
- Then, he went 1975 to The University of New Mexico, to work with Cleve Moler, towards a PhD degree in Mathematics and working at Los Alamos Natl Lab. He got his PhD degree in Appl. Mathematics from the University of New Mexico in 1980.
- At Los Alamos he got in touch with the Cray 1, the first **supercomputer**, and started a new project, **LINPACK**, the Linear Algebra Package. His PhD dissertation was on improving the accuracy of eigenvalue computations.

Short Summary of Dongarra's Career

- Citations Google Scholar: **115,267**, h-index **132**, i-10 index **922**, MR Erdos Number = 3
- Contributions in numerical algorithms in linear algebra, parallel computing, the use of advanced-computer architectures, programming methodology, and tools for parallel computers
- Open source software packages and systems: EISPACK, LINPACK, BLAS, LAPACK, ScaLAPACK, Netlib, PVM, MPI, NetSolve, Top500, ATLAS, and PAPI
- Prizes and Awards:
  - IEEE Sid Fernbach Award (2004) for his contributions in the application of high performance computers using innovative approaches;
  - IEEE Medal of Excellence in Scalable Computing (2008)
  - SIAM Special Interest Group on Supercomputing's award for Career Achievement (2010)
  - IEEE Charles Babbage Award (2011)
  - ACM/IEEE Ken Kennedy Awar (2013) for his leadership in designing and promoting standards for mathematical software used to solve numerical problems common to high performance computing
  - SIAM/ACM Prize in Computational Science and Engineering (2019)
  - IEEE Computer Pioneer Award (2020) for leadership in the area of high-performance mathematical software
  - ACM A.M. Turing Award (2022) for pioneering contributions to numerical algorithms and software that have driven decades of extraordinary progress in computing performance and applications.
- He is a Fellow of the AAAS, ACM, IEEE, and SIAM and a Foreign Fellow of the British Royal Society and a Member of the US National Academy of Engineering.

What Dongarra has done to earn the ACM Turing Award?

Who else can say this better than Dongarra himself?

https://youtu.be/Oe9LRKoE6L0

Packages of Mathematical Software

The Basic Linear Algebra Subprograms (BLAS)

- Set of standard, efficient and portable routines to compute vector-vector (BLAS1), matrix-vector (BLAS2), and matrix-matrix (BLAS3) kernel operations
- BLAS1 designed in the late 70s for vector computers by Lawson & Henson from JPL. It includes dot products, L2-norm computations, vector updates, vector multiply & add (axpy). Uses scalar optimization and vectorization techniques (e.g, loop unrolling).
- BLAS2 includes matrix-vector operations, matvec multiplication for various matrix formats (dense, band, tridiagonal …), transpose, rank-1 updates, etc,
- BLAS3 supports matrix-matrix operations, such as matrix-matrix multiply and add, etc
- BLAS2 & BLAS3 critical to gaining performance on machines that have a memory hierarchy architecture

Dot product source code (blas/sdot.f)

```
*
*  Definition:
*  ===========
*
*       REAL FUNCTION SDOT(N,SX,INCX,SY,INCY)
*
*       .. Scalar Arguments ..
*       INTEGER INCX,INCY,N
*       ..
*       .. Array Arguments ..
*       REAL SX(*),SY(*)
*       ..
*
*
*> \par Purpose:
*  =============
*>
*> \verbatim
*>
*>    SDOT forms the dot product of two vectors.
*>    uses unrolled loops for increments equal to one.
*> \endverbatim
*
*  Arguments:
*  ==========
*
*> \param[in] N
*
…… comment comment comment ……...
**  Authors:
*  ========
*
*> \author Univ. of Tennessee
*> \author Univ. of California Berkeley
*> \author Univ. of Colorado Denver
*> \author NAG Ltd.
*
*> \ingroup single_blas_level1
*
*> \par Further Details:
*  =====================
*>
*> \verbatim
*>
*>     jack dongarra, linpack, 3/11/78.
*>     modified 12/3/93, array(1) declarations changed to array(*)
*> \endverbatim
```

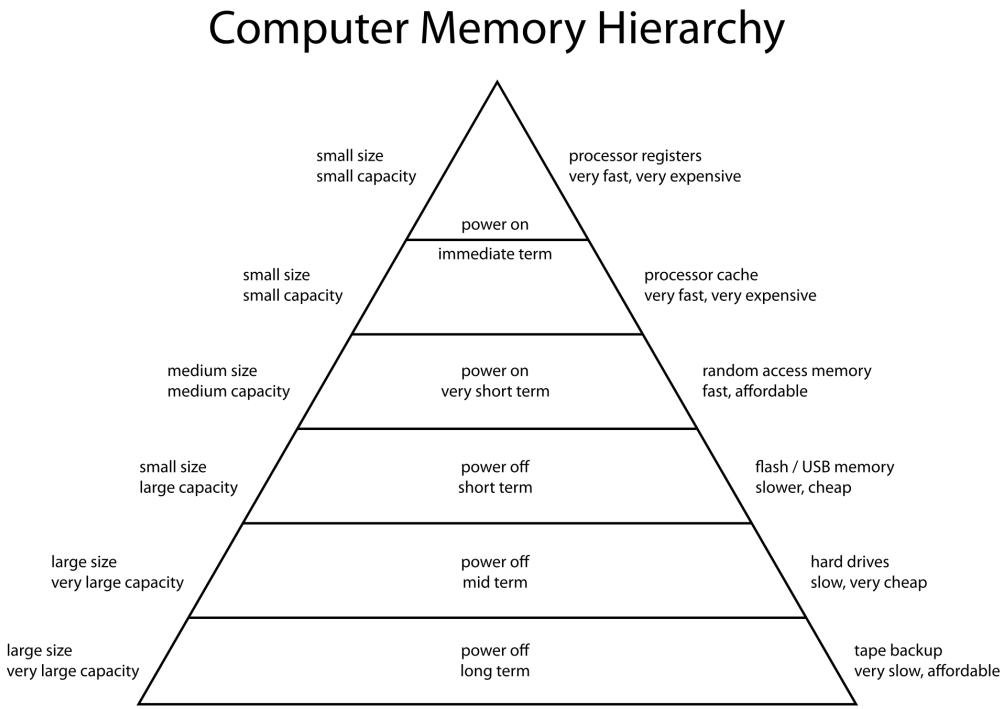
```
*
*        code for both increments equal to 1
*
*
*        clean-up loop
*
         M = MOD(N,5)
         IF (M.NE.0) THEN
            DO I = 1,M
               STEMP = STEMP + SX(I)*SY(I)
            END DO
            IF (N.LT.5) THEN
               SDOT=STEMP
            RETURN
            END IF
         END IF
         MP1 = M + 1
         DO I = MP1,N,5
          STEMP = STEMP + SX(I)*SY(I) + SX(I+1)*SY(I+1) +
     $            SX(I+2)*SY(I+2) + SX(I+3)*SY(I+3) +
SX(I+4)*SY(I+4)
         END DO
      ELSE
*
*        code for unequal increments or equal increments
*          not equal to 1
*
         IX = 1
         IY = 1
         IF (INCX.LT.0) IX = (-N+1)*INCX + 1
         IF (INCY.LT.0) IY = (-N+1)*INCY + 1
         DO I = 1,N
            STEMP = STEMP + SX(IX)*SY(IY)
            IX = IX + INCX
            IY = IY + INCY
         END DO
      END IF
      SDOT = STEMP
      RETURN
```

The Cray 1 supercomputer

In the Cray 1 vector operations are one instruction

Today's computers still can benefit from loop unrolling!

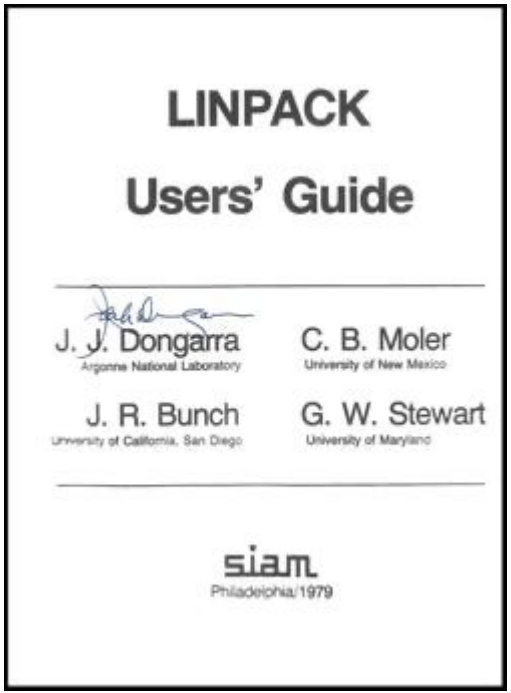
Memory Hierarchy and BLAS2 & BLAS3




```
> \brief \b SGEMV
*
*  =========== DOCUMENTATION ===========
*
* Online html documentation available at
*            http://www.netlib.org/lapack/explore-html/
*
*  Definition:
*  ===========
*
*       SUBROUTINE SGEMV(TRANS,M,N,ALPHA,A,LDA,X,INCX,BETA,Y,INCY)
*
*       .. Scalar Arguments ..
*       REAL ALPHA,BETA
*       INTEGER INCX,INCY,LDA,M,N
*       CHARACTER TRANS
*       ..
*       .. Array Arguments ..
*       REAL A(LDA,*),X(*),Y(*)
*       ..
*
*
*> \par Purpose:
*  =============
*>
*> \verbatim
*>
*> SGEMV  performs one of the matrix-vector operations
*>
*>    y := alpha*A*x + beta*y,   or   y := alpha*A**T*x + beta*y,
*>
*> where alpha and beta are scalars, x and y are vectors and A is an
*> m by n matrix.
*> \endverbatim
*>  Level 2 Blas routine.
*>  -- Written on 22-October-1986.
*>     Jack Dongarra, Argonne National Lab.
*>     Jeremy Du Croz, Nag Central Office.
*>     Sven Hammarling, Nag Central Office.
*>     Richard Hanson, Sandia National Labs.
*> \endverbatim
```

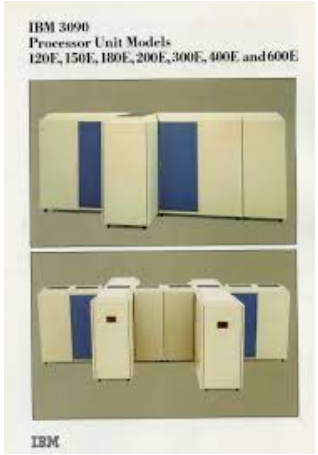
The Linear Algebra Package - LINPACK

- A package of subroutines for solving systems of linear equations (Ax=b) and linear least-square problems;
- The package solves linear systems whose matrices are general, banded, symmetric indefinite, symmetric positive definite, triangular, and tridiagonal square. In addition, the package computes the QR and SVD of rectangular matrices and applies them to least-squares problems. LINPACK uses column-oriented algorithms to increase efficiency by preserving locality of references.
- All of the inner loops are done by calls to the BLAS
- In 1981, B. Parlett from UC Berkeley: “It may seem strange that SIAM should publish and review a users' guide for a set of Fortran programs. Yet history will show that SIAM is thereby helping to foster a new aspect of technology which currently rejoices in a name mathematical software. There is as yet no satisfying characterization of this activity, but it certainly concerns the strong effect that a computer system has on the efficiency of a program.”



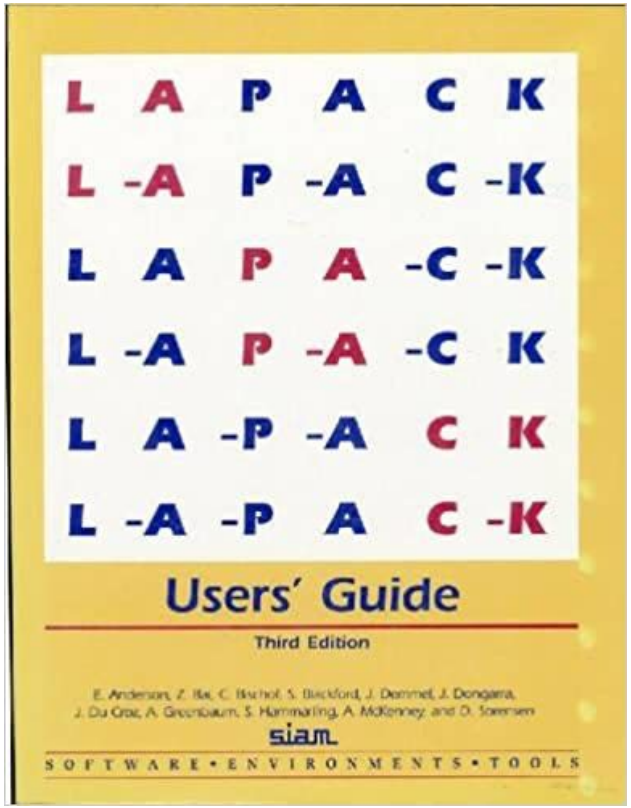
Shared Memory Supercomputers: Cray YMP/IBM 3090/NEC SX

- The next step for vector computers were to evolve for multiple vector processors sharing a single memory.
- The most powerful examples at the time were the Cray YMP, the IBM 3090, and the NEC SX



The LAPACK Project

- The LAPACK was an effort to take EISPACK & LINPACK to work in shared memory machines, the most powerful computers in late 80s and early 90s.
- LAPACK was structured on top of the BLAS. It was focused whenever possible in terms of Level 3 BLAS, matrix-matrix operations, to expose the high levels of performance that one can get out through matrix multiplication and its related operations.
- From the moment it was released it was the standard for numerical software.



Q: Where you can find BLAS and LINPACK today?

A: Pretty much everywhere!

- Routines can be downloaded from netlib.org (FORTRAN, C)
- They are present in compilers: GNU, Intel (MKL), PGI, NVIDIA
- There are GPU versions: NVBLAS, cuBLAS
- TensorFlow wraps BLAS & LINPACK calls for optimizing its kernel
- They are in Python: scipy, numpy, scikit-learn
- They can be used in Matlab, Octave, R, Julia

Portability and performance guaranteed!

Message Passing Interface

Parallel Architectures Fundamentals

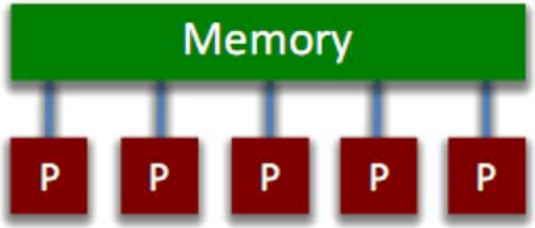
Instruction stream

Data stream

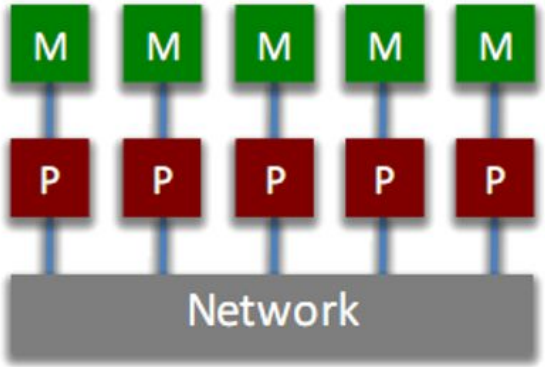
SISD Traditional Von Neuman Single CPU computer	**MISD** Pipelined computers
SIMD Vector Processors Fine grained data parallel computers	**MIMD** Multicomputers Multiprocessors

Finn's classification

Shared memory

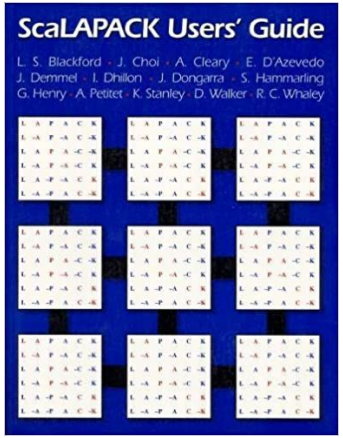


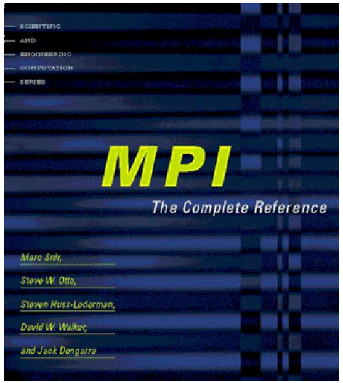
Distributed memory



Dongarra's contributions to message passing

- In 1991, Dongarra & Demmel (UCB) got a DARPA grant that led to ScaLAPACK, the Scalable Linear Algebra Package
- Idea was to take build an abstraction to EISPACK & LINPACK, ensuring portability and efficiency in the new distributed memory machines out there.
- ScaLAPACK uses parallel BLAS, a set of routines developed using the message passing library **MPI**, which D. was a developer.
- ScaLAPACK also used PVM, a library developed by D. and his team at Oak Ridge, together with Emory and the University of Tennessee.
- It turns out that **MPI** became today's **de facto industrial standard** (https://www.mpi-forum.org).





The LINPACK Benchmark and the TOP500 list

LINPACK Benchmark Origins

- According to D., Luszczek & Petitet[1], the LINPACK benchmark was originally designed to assist users by providing information on the execution times required to solve a system of linear equations of size *N*=100 by LU factorization using double precision (float64). The 1979 report contains data for 23 computers.
- The LINPACK has evolved to:

Table I. Overview of nomenclature and rules for the LINPACK suite of benchmarks. The LINPACK 1000 Benchmark is also known as Toward Peak Performance (TPP) or Best Effort. The Highly-Parallel LINPACK (HPLinpack) Benchmark is also known as the $N \times N$ LINPACK Benchmark or High Parallel Computing (HPC).

Benchmark name	Matrix dimension	Optimizations allowed	Parallel processing
LINPACK 100	100	Compiler	Compiler parallelization possible
LINPACK 1000	1000	Manual	Multiprocessor implementations allowed
LINPACK Parallel	1000	Manual	Yes
HPLinpack	Arbitrary	Manual	Yes

[1]Dongarra, Jack J., Piotr Luszczek, and Antoine Petitet. "The LINPACK benchmark: past, present and future." Concurrency and Computation: practice and experience 15.9 (2003): 803-820.

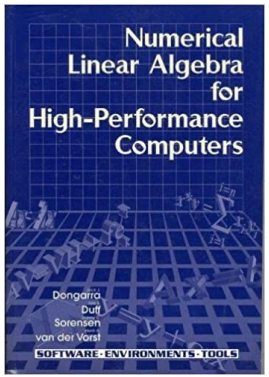
The HPL benchmark rules

- The HPL benchmark has a set of rules:
  - Solve a nxn dense linear system of equations by LU factorization with partial pivoting in parallel allowing the problem size n to vary, measure the the execution time. Ideally, the size n should be made large enough so that asymptotic performance is observed.
  - Compute the floating-point execution rate using the #operations: $2n^3/3 + 2n^2$
  - Compute and report solution residual as: $||Ax-b|| / (||A|| ||b||)$
- Report the following quantities:
  - R_{max}, the performance in Gflop/s for the largest problem run
  - N_{max}, the size of the largest problem run
  - $N_{1/2}$, the size where half R_{max} was achieved
  - R_{peak}, the theoretical peak performance in Gflop/s

What is a supercomputer (high performance computer)?

Definition (Dongarra et al, 1998):

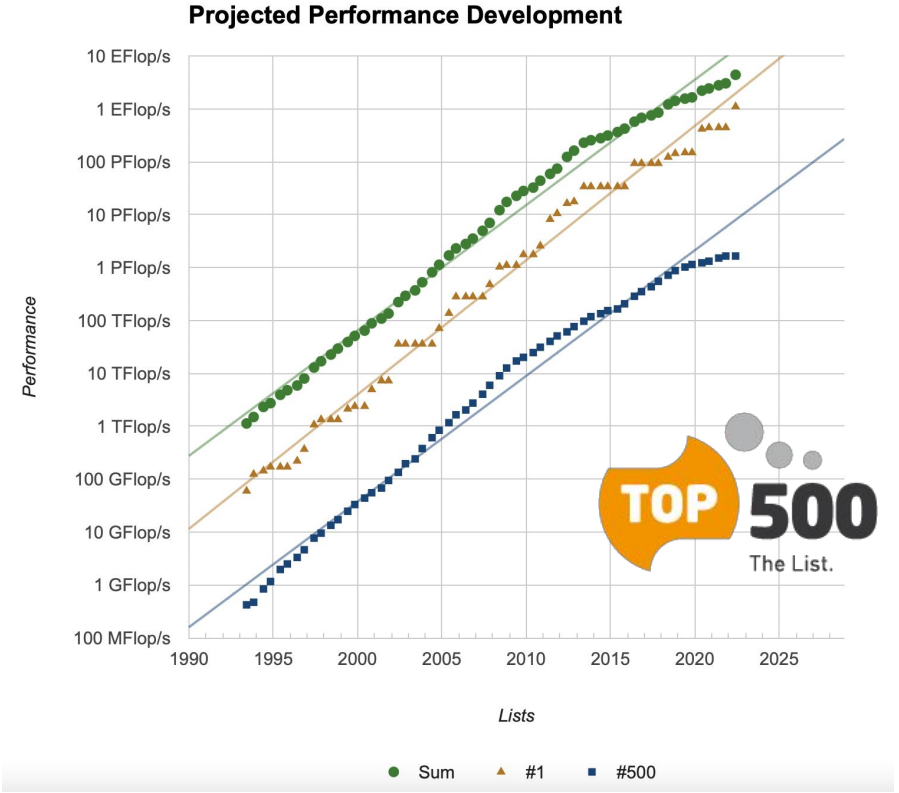
Supercomputers are the fastest and most powerful general purpose scientific computing systems available at any given time.



HPL outputs

Fastest: R_{max}

Most powerful: largest n



The 2022 Breakthrough: we're in the Exaflop era!

#1 Machine Oak Ridge Frontier, USA

Rmax	1,102.00 PFlop/s
Rpeak	1,685.65 PFlop/s
Nmax	24,440,832
Power	21,100.00 kW
#cores	8,730,112 AMD

https://youtu.be/etVzy1z_Ptg

1 EFlop/s = 10^{18} Flop/s =

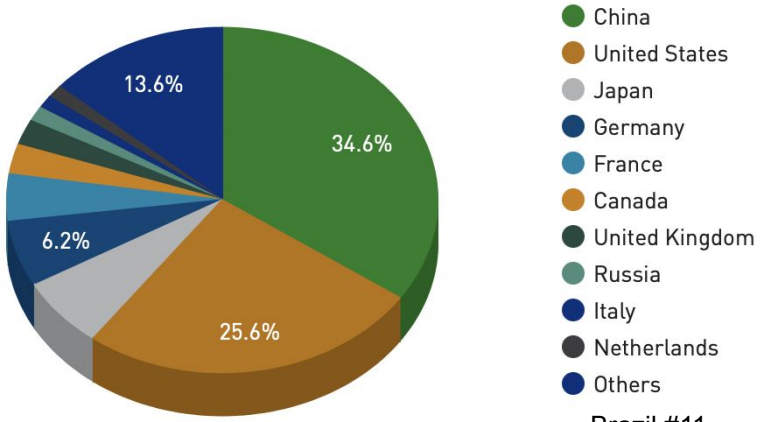
1 000 000 000 000 000 000 Flop/s

The TOP500 list: a technological advancement indicator

Rank	Site	System	Cores	Rmax Tflop/s	Rpeak Tflop/s	Power kW
1	DOE/SC/Oak Ridge National Laboratory **United States**	Frontier - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11	8,730,112	1,102.00	1,685.65	21,100
2	RIKEN Center for Computational Science **Japan**	Supercomputer Fugaku , A64FX 48C 2.2GHz, Tofu interconnect D Fujitsu	7,630,848	442.01	537.21	29,899
3	EuroHPC/CSC **Finland**	LUMI - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11	1,110,144	151.90	214.35	2,942
4	DOE/SC/Oak Ridge National Laboratory **United States**	Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband	2,414,592	148.60	200.79	10,096
5	DOE/NNSA/LLNL **United States**	Sierra - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband	1,572,480	94.64	125.71	7,438

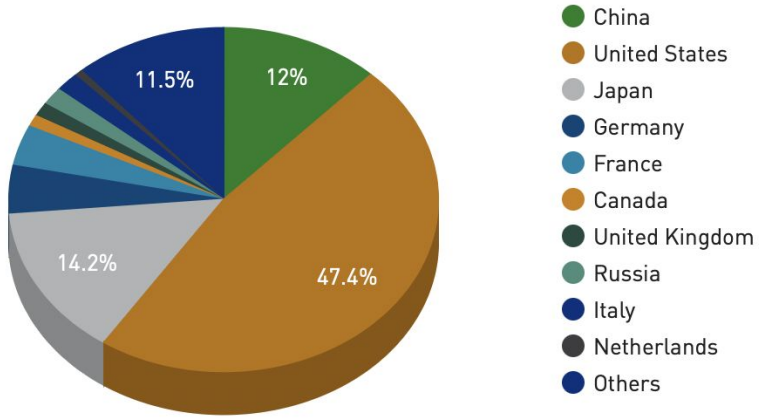
Where are the TOP500 systems?

Countries System Share



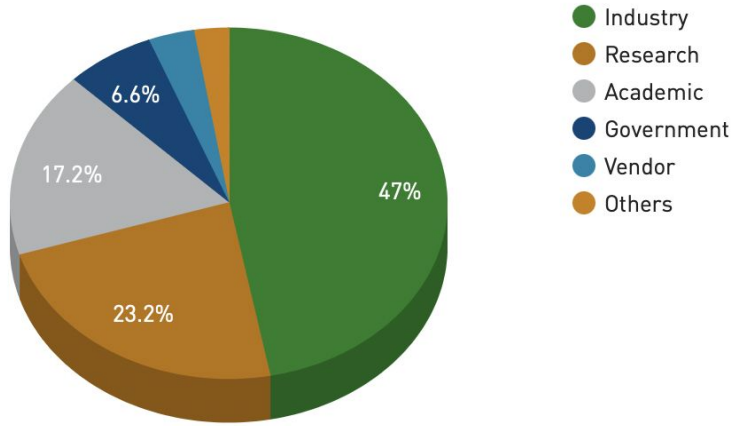
Brazil #11

Countries Performance Share

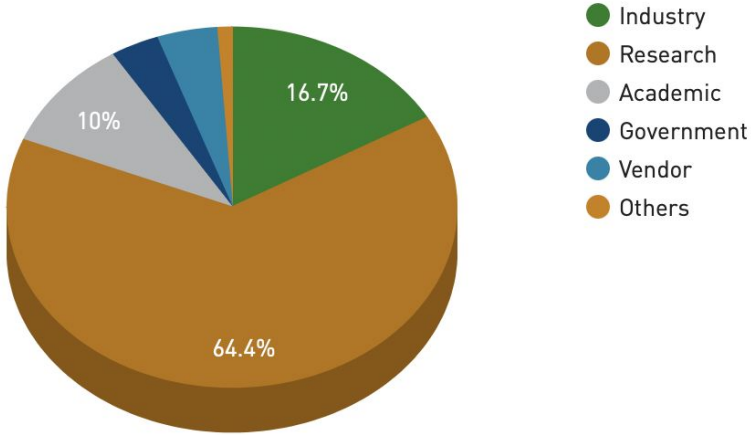


In what the TOP500 systems are used?

Segments System Share



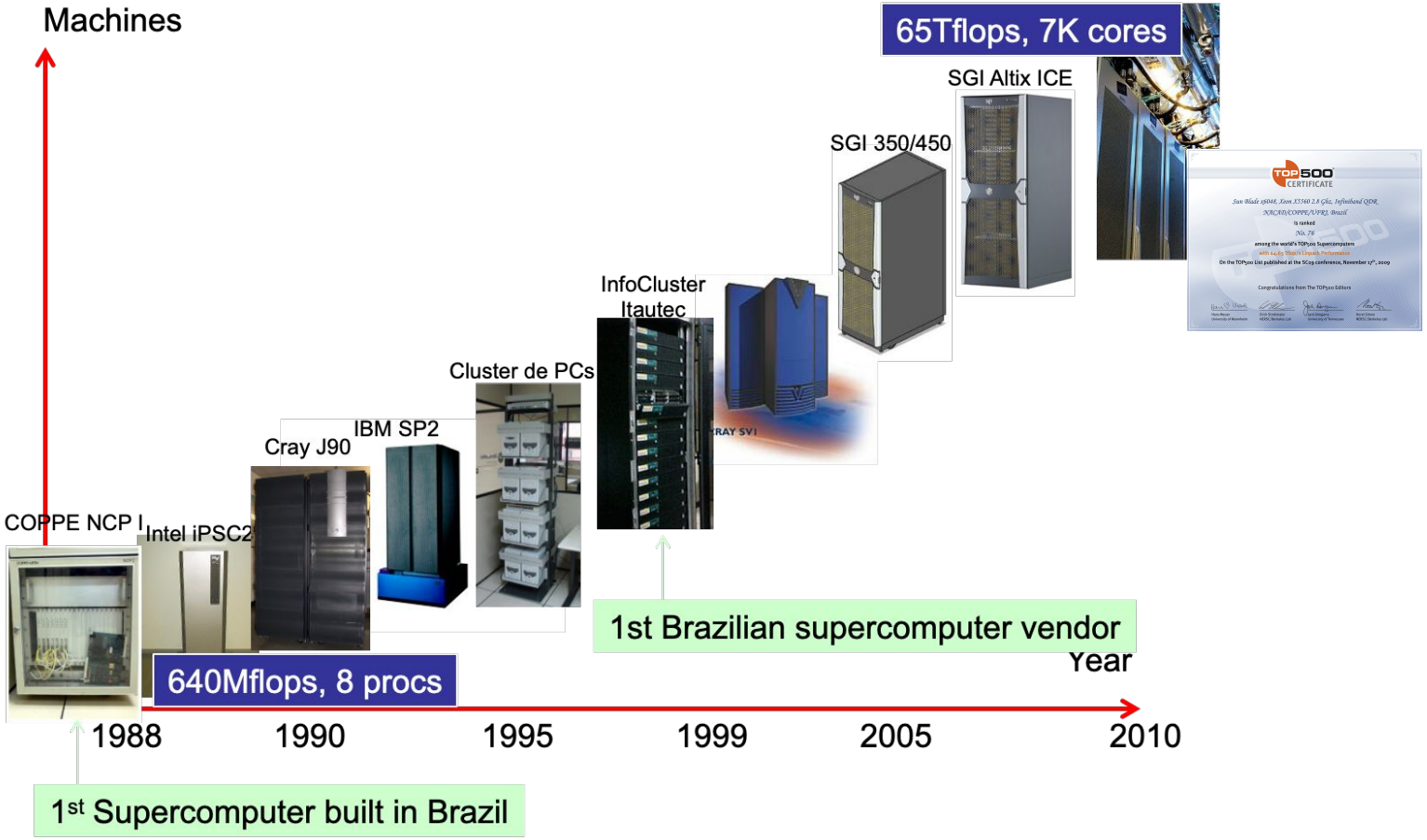
Segments Performance Share



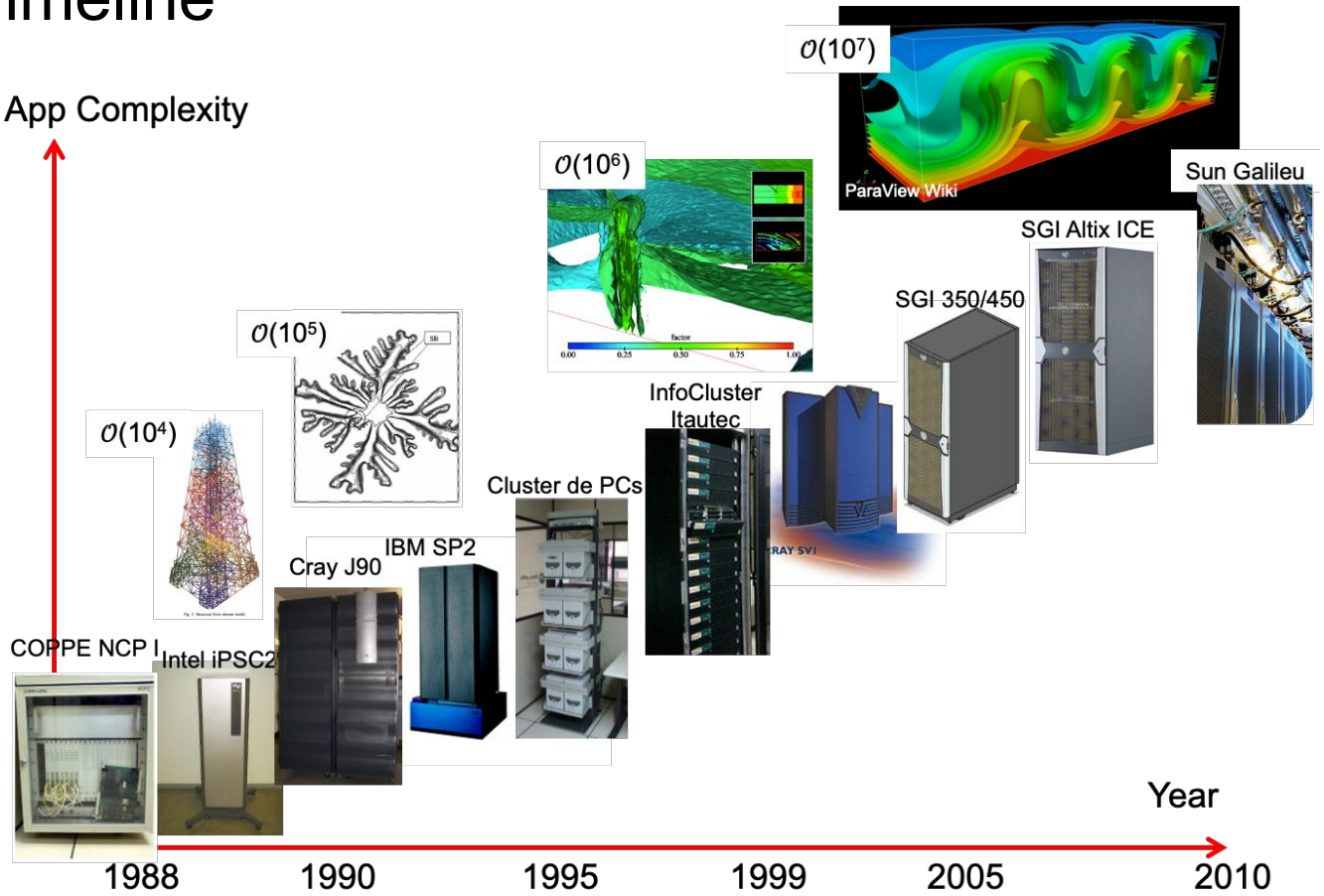
Brazil in the TOP500: #11 in systems share

Rank	System	Cores	Rmax (Tflop/s)	Rpeak (Tflop/s)	Power (kW)
60	Dragão - Supermicro SYS-4029GP-TVRT, Xeon Gold 6230R 26C 2.1GHz, NVIDIA Tesla V100, Infiniband EDR, Atos, **PETROBRAS**	188,224	8.98	14.01	943
116	Atlas - Bull 4029GP-TVRT, Xeon Gold 6240 18C 2.6GHz, NVIDIA Tesla V100, Infiniband EDR, Atos, **PETROBRAS**	91,936	4.38	8.85	547
137	IARA - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100 SXM4 40 GB, Infiniband, Nvidia, **SiDi**	24,800	3.66	4.13	
161	Fênix - Bull 4029GP-TVRT, Xeon Gold 5122 4C 3.6GHz, NVIDIA Tesla V100, Infiniband EDR, Atos, **PETROBRAS**	60,480	3.16	5.37	390
352	A16A - ThinkSystem C0366, Xeon Gold 6252 24C 2.1GHz, 100G Ethernet, Lenovo **Software Company MBZ**	61,440	2.09	4.13	
424	Santos Dumont - Bull Sequana X1000, Xeon Gold 6252 24C 2.1GHz, Mellanox InfiniBand EDR, NVIDIA Tesla V100 SXM2, Atos, **LNCC**	33,856	1.85	2.73	

COPPE's HPC Timeline

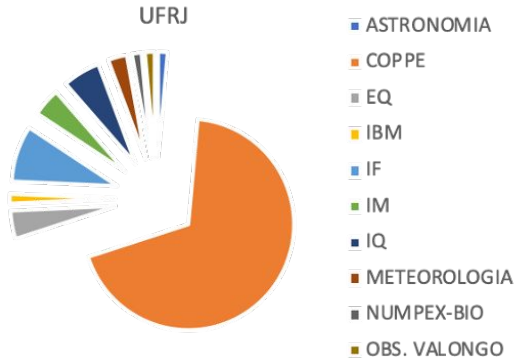


Apps Timeline



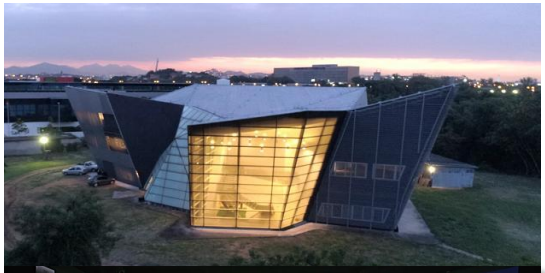
Lobo Carneiro Supercomputer

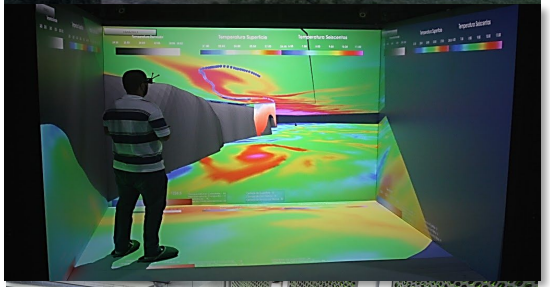
COPPE's machine: Lobo Carneiro
253 Compute Nodes 6072 Cores (506 Processors Intel Xeon Haswell E5-2670V3 12-Cores 2.3GHz,16.192GB Mem DDR4, 193 Tflop/s



Since 2016: 150 projects, 400 users, 1M jobs
COPPE: **30 Projects** directly related to Petrobras

COPPE Digital Hub





- ***NACAD***: *High Performance Computing Center*
- ***NTT:*** *Technology Transfer Center*
- ***LAMCE:*** *Laboratory of Computational Methods in Engineering*
- ***LRAP:*** *Advanced Petroleum Recovery Laboratory*
- ***LPS:*** *Signal Processing Laboratory*
- ***SMT:*** *Signals Laboratory, Multimedia and Telecommunications*
- ***GSCAR:*** *Simulation and Control Group in Automation and Robotics*
- ***Lab3D:*** *Virtual Reality Lab*
- ***Lab2M:*** *Multidisciplinary Modeling Laboratory*
- ***LENS:*** *Software Engineering Laboratory*
- ***LADES:*** *Solutions for Process Control and Optimization Lab*
- ***ERGOPROJ****: Lab Ergonomics & Projects*

My recollections of J. Dongarra

- Introduced to D. by H. Simon at the SIAM Meeting on Parallel and Distributed Computing, Minneapolis, March 15, 1997, when I was visiting the AHPRC, U. Minneapolis. I presented a paper there.
- Meet him again in 1999, at the SIAM Conference on Parallel Processing for Scientific Computing, March, San Antonio, TX, USA, where I also presented a paper.
- He came to Brazil as an invited speaker at SBAC-PAD, in Vitoria/ES - Brazil - October 28-30, 2002 (yet another paper!).
- We meet again in Brazil at the NUG workshop, Angra dos Reis, 2003.
- Since then we met regularly at SIAM and SC meetings. Last time we met in-person was in 2019, at the SIAM CSE Meeting, Spokane, WA. At that time he was awarded the SIAM/ACM Prize in Computational Science.

Two important moments

Receiving from D. the #1LA Supercomputer Award, SC2009, Portland, OR, Galileu Supercomputer, COPPE

With L. Gesenhues, former student (with F. Rochinha) who won the prestigious 2018 ACM-IEEE CS George Michael Memorial HPC Fellowships, SC2017, Denver, CO, USA (D. in the committee)

Discussion: back to where we began

- Who is Jack Dongarra?

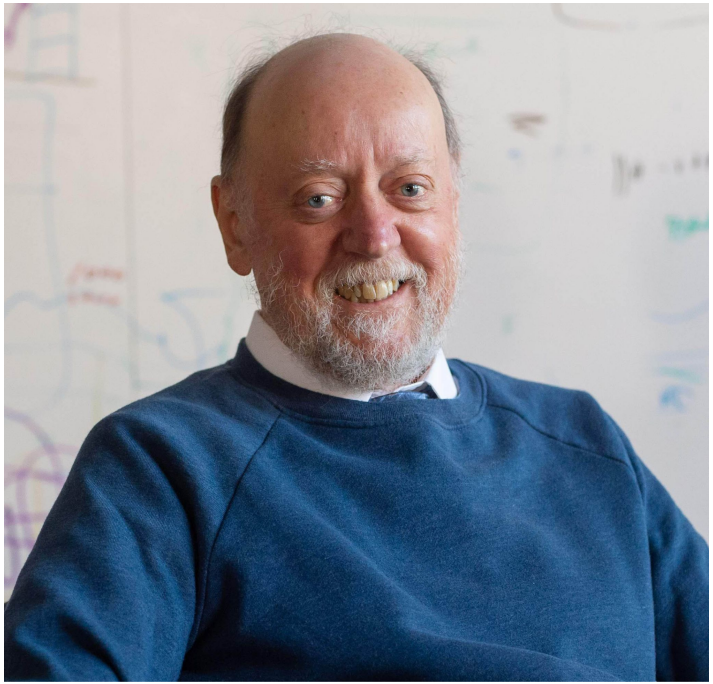
One of the most important living computational scientists. His work helped to define the field.

- What Dongarra has done to earn the ACM Turing Award?

Contributed to the fields of mathematical software, parallel computing and supercomputing.

- Why this is important?

Supercomputers are everywhere! They are parallel machines, they need to exchange messages. Mathematical kernels are pervasive in science and engineering.



Congrats, Jack!