THE EVOLUTION OF HIGH-PERFORMANCE SYSTEMS:
FROM HPC TO BIG DATA TO DEEP LEARNING

Marco Aldinucci
Computer Science Department, University of Torino

Head of Parallel Computing group, coordinator of HPC4AI,
national delegate at EuroHPC JU
DATA LOSES ITS OPERATIONAL VALUE IN A SHORT TIME

• Partition the atmosphere 1x1x1 Km up to 10 Km altitude (10 cells). Earth: ~ 5 x 10^9 cells

• Let us suppose 1 cell needs 200 Floating Point Ops. I.e. 10^12 Ops for each step

• A 7-days forecast with 1 minute time step on a 10GFLOPS CPU needs 10^7 secs, i.e. 10 days
  • Es. Intel Core i7-7500U

• To compute it in 5 mins you need a 35 TFLOPS CPU
  • 3500x (fastest CPU today, ~200GFLOPS)
Self-Driving Cars Can Handle Neither Rain nor Sleet nor Snow

- To help autonomous vehicles solve inclement conditions, WaveSense will sell a sensor that can see below the ground.

By Kyle Stock

Hyperdrive
PROGRAMMING MODEL: STENCIL

\[
Cell_{(x,y)}^{T+1} = P_1 Cell_{(x,y-1)}^T + P_2 Cell_{(x-1,y)}^T + P_3 Cell_{(x+1,y)}^T + P_4 Cell_{(x,y+1)}^T
\]
PROGRAMMING MODELS AND APPLICATIONS

HPC

- Locally synchronous data parallelism
  - Stencil
  - Scalable and well-understood
  - Simulations, finite elements, AMR, HPLinpack, …
  - Physics, chemistry, engineering, …

- Globally synchronous data parallelism
  - BSP, PGAS, …
  - Barrier = not scalable

- Stream & task parallel
  - Compositional = good, but not scalable

M. Aldinucci et al. “A Parallel Pattern for Iterative Stencil + Reduce,” Journal of Supercomputing, 2018
HPC APPLICATIONS

L’OMA SEMPER FAIT PAREI… (SARA PÀ PERICULUS)

Number of applications using a given programming paradigm in the set of 30 candidate exascale applications of the USA DoE Exascale Computing Project (ECP).
PLATFORMS

• Multi-level clusters
  • Multicore + GPUs

• Data movement
  • Optimised for CPU-bound applications, I/O-bound applications problematic

• High-speed network (e.g. fat-tree IB 100-400 Gb/s)
  • Efficient synchronisation is key for scalability

• Managed with job queue (PBS, SLURM)
  • Sometime bare metal virtualisation: singularity, kubernetes, …Occam@UNITO
Advantages: Interactive, virtual farms + queue, user-defined configuration

More importantly… we designed it!
Pre-exascale general system specifications

- Applicants must describe how the following general system specifications will be met, for both the EuroHPC supercomputer and the site.

- The hosting entity will host a supercomputer with the following requirements:
  - A capability computing system, with an aggregated performance level capable of executing at least 200 Petaflops (sustained performance measured using linpack benchmark)
  - Covering the needs (including substantial performance increase) of a wide range of key/grand challenge applications that demonstrably require large systems that are precursors of exascale capability computing.
  - A total power consumption of no more than 15 MW for the hosting of the EuroHPC supercomputer
"The most damaging phrase in the language is ‘L’OMA SEMPER FAIT PAREI’" (we’ve always done it this way!)

Grace Hopper
From Wikipedia, the free encyclopedia

Grace Brewster Murray Hopper (née Murray; December 9, 1906 – January 1, 1992) was an American computer scientist and United States Navy rear admiral.[1] One of the first programmers of the Harvard Mark I computer, she was a pioneer of computer programming who invented one of the first compiler related tools. She popularized the idea of machine-independent programming languages, which led to the development of COBOL, an early high-level programming language still in use today.

Hopper attempted to enlist in the Navy during World War II but was rejected because she was 34 years old. She joined the Navy Reserve. Hopper began her computing career in 1944 when she worked on the Mark I team led by Howard Aiken. In 1949, she joined the Eckert–Mauchly Computer Corporation, a part of the team that developed the UNIVAC I computer. At Eckert–Mauchly she began developing the UNIVAC I compiler. She believed that a programming language based on English was possible. Her compiler converted terms into machine code understood by computers. By 1952, Hopper had finished her program linker (later called a compiler), which was written for the A-0 System.[3][4][5]

Eckert–Mauchly chose Hopper to lead their department for automatic programming, and she led the development of some of the first compiled languages like FLOW-MATIC. In 1959, she participated in the CODASYL committee, which consulted Hopper to guide them in creating a machine-independent programming language. She worked on the COBOL language, which was inspired by her idea of a language being based on English words, and she retired from the Naval Reserve, but in 1967, the Navy recalled her to active duty. She retired in 1986 and found work as a consultant for the Digital Equipment Corporation, sharing her computing experiences.

Her accomplishments and her naval rank, she was sometimes referred to as "Amazing Grace".[6][7] The Navy Arleigh Burke-class guided-missile destroyer USS Hopper was named for her, as was the Cray XE6 "Hopper" supercomputer at NERSC.[8] During her lifetime, Hopper was awarded 40 honorary degrees from various universities around the world.

Rear Admiral Grace M. Hopper, 1984
Born December 9, 1906
New York City, New York, U.S.
Died January 1, 1992 (aged 85)
Arlington, Virginia, U.S.
Other names "Amazing Grace", "Grandma COBOL"
Alma mater Yale University
Military career
PARALLEL COMPUTING GROUP FUNDING PERSPECTIVE (ALPHA@UNITO)

- **ParaPhrase** (EC-STREP, 7th FP): Parallel Patterns for Adaptive Heterogeneous Multicore Systems (2011, 42 months, total cost 4.2M €).
- **CINA** (MIUR PRIN): Compositionality, Interaction, Negotiation, Autonomicity for the future ICT society (2013, 36 months).
- **REPARA** (EC-STREP, 7th FP): Reengineering and Enabling Performance And poweR of Applications (2013, 36 months, total cost 3.5M €).
- **C3S**: Competence Center on Scientific Computing (2014, Compagnia di San Paolo, founding 900K €).
- **cHiPSet** (EC-COST Action IC1406): High-Performance Modelling and Simulation for Big Data Applications (2015, 48 months, total cost 500K).
- **OptiBike** (EU I4MS): Robust Lightweight Composite Bicycle design and optimization, an experiment of EU i4MS Fortissimo2 project (2017, 24 months, total cost 230K €).
- **HPC4AI** (Regione Piemonte, INFRA_P): Turin’s centre in High-Performance Computing for Artificial Intelligence (2018, 24 months, total cost 4.5M €).
- **MnemoComputing** (Compagnia di SanPaolo): Components for Processing In Memory (2019, 24 months, total cost 70K€)
- **PDEvolve** Deep Learning + HPC (ICT-11-b - under evaluation)
- **ExaTrain** Deep Learning + HPC (Marie Curie - under evaluation)
REINFORCED DEEP LEARNING

It might look goofy ...
FLOPS

- **NVIDIA V100**
  - 32 bit ~15TFLOPS
  - 5120 cores + 640 tensor cores
  - 300W, cost: ~7K€

- **Intel E5-2699v4**
  - 64 bit ~200GFLOPS
  - 22 cores HT + AVX2
  - 150W, cost: ~4K€

- **Marconi**
  - 64 bit ~10PFLOPS
  - 350,000 KNL 68 cores
  - 3MW, cost: 30M€

---

**Min training time**

- AlphaZero ~36 years
- AlphaZero ~684 years
- AlphaZero ~1 hour
The new Turing GPU
Less than 2500€

NVIDIA TESLA T4 SPECIFICATIONS

Performance
- Turing Tensor Cores: 320
- Mixed-Compute Cores: 2,560
- Single Precision Performance (FP32): 8.1 TFLOPS
- Mixed Precision (FP16/FP32): 65 TFLOPS
- INT8 Precision: 130 INT8 TOPS
- INT4 Precision: 260 INT4 TOPS

Interconnect
- GEN3
- x16 PCIe

Memory
- Capacity: 16 GB DDR4
- Bandwidth: 320+ GB/s

Power
- 70 watts

Figure 4. Turing TU102/TU104/TU106 Streaming Multiprocessor (SM)
THE NEW TURING GPU
LESS THAN 2500€

CONCURRENT EXECUTION

Per 100 FP instructions,
average 36 INT PIPE instructions
(ie ladd, select, fp min/max, compare etc)

Figure 4. Turing TU102/TU104/TU106 Streaming Multiprocessor (SM)
From FP32 to FP16 to INT8, as well as INT4
Ternary Neural Networks for Resource-Efficient AI Applications

Hande Alemdar*, Vincent Leroy*, Adrien Provost-Bouclé†, Frédéric Pétrot‡
*Université Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France
†Univ. Grenoble Alpes, CNRS, Grenoble INP, TIMA, F-38000 Grenoble, France
Email: name.surname@univ-grenoble-alpes.fr

Abstract—The computation and storage requirements for Deep Neural Networks (DNNs) are usually high. This issue limits their deployability on ubiquitous computing devices such as smart phones, wearables and autonomous drones. In this paper, we propose ternary neural networks (TNNs) in order to make deep learning more resource-efficient. We train these TNNs using a teacher-student approach based on a novel, layer-wise greedy learning method. Thanks to our two-stage training procedure, the teacher network is still benefiting all state-of-the-art techniques such as dropout and batch normalization to increase accuracy and reduce training time. Using only ternary weights and activations, the student network learns to mimic the behavior of its teacher network without using any multiplication. Unlike in [1-3], our purpose-built hardware architecture for TNNs and implement it on FPGA and ASIC. We evaluate TNNs on several benchmark datasets and demonstrate up to 3× better energy efficiency with respect to the state of the art while also improving accuracy.

TABLE I

<table>
<thead>
<tr>
<th>TERNARY NEURAL NETWORK DEFINITIONS FOR A SINGLE NEURON</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher network</td>
<td>Student network</td>
<td></td>
</tr>
<tr>
<td>Weights</td>
<td>$W_i^t = {w_j}, w_j \in \mathbb{R}$</td>
<td>$W_i^s = {w_j}, w_j \in {-1, 0, 1}$</td>
</tr>
<tr>
<td>Bias</td>
<td>$b_i^t \in \mathbb{R}$</td>
<td>$b_i^s \in \mathbb{Z}$</td>
</tr>
<tr>
<td>Transfer Function</td>
<td>$y_i^t = W_i^t x + b_i^t$</td>
<td>$y_i^s = W_i^s x$</td>
</tr>
<tr>
<td>Act. Fun</td>
<td>$n_i^t = \begin{cases} -1 \text{ with prob. } -\rho \text{ if } \rho &lt; 0 \ 1 \text{ with prob. } \rho \text{ if } \rho &gt; 0 \ 0 \text{ otherwise} \end{cases}$</td>
<td>$n_i^s = \begin{cases} -1 \text{ if } y_i^s &lt; b_i^s \ 1 \text{ if } y_i^s &gt; b_i^s \ 0 \text{ otherwise} \end{cases}$</td>
</tr>
</tbody>
</table>

where $\rho = \tanh(y_i), \rho \in (-1, 1)$

without extensive data augmentation. TNN’s error rate on MNIST is 1.67% with a single 3-layer MLP with 750 neurons in each layer. Bitwise NNs [10] with 1624 neurons in 3 layers achieves a slightly better performance. TNN with an architecture that has similar size to Bitwise NN is worse due to over-fitting. Since TNN selects a different sparsity level for each neuron, it can perform better on smaller networks, and larger networks can over-fit on MNIST. Bitwise NN’s global sparsity parameter has a better regularization effect on MNIST.

TABLE IV

| CLASSIFICATION PERFORMANCE - ERROR RATES (%) |
|---|---|---|---|---|
| Fully Discretized | TNN (This Work) | ThunkNet [11, 12] | MNIST | CIFAR10 |
| Ternary Network | TNN (This Work) | 7.90 | 14.59 | 3.54 | 3.95 |
| Byte Normalized | 1.35 |
| Partially Discretized | Relaxed NN [4] | 0.96 | 10.15 | 2.53 |
| Binarized NN | 1.29 | 9.43 | 2.20 |
| XNOR-Net [16] | 1.15 | 12.61 | 2.42 |
| TC [14] | 0.65 | 7.44 |
| BC [15] | 2.20 | 9.48 | 2.40 |
| SWB (4x) | 2.20 | 9.48 |
| DFTxNet [17] | 2.20 | 9.48 |

A. Hardware Architecture

Figure 3 outlines the hardware architecture of a fully-connected layer in a multi-layer NN. The design forms a pipeline that corresponds to the sequence of NN processing steps. For efficiency reasons, the number of layers and the maximum layer dimensions (input size and number of neurons) are decided at synthesis time. For a given NN architecture, the design is still user-programmable: each NN layer contains a memory that can be programmed at run-time with neuron weights or output termination thresholds $W$ and $V$. As seen in the previous experiments of Section IV, a given NN architecture can be reused for different datasets with success. Ternary values are represented with 2 bits using usual two’s complement encoding. That way, the compute part of each neuron is reduced to just one integer adder/subtractor and one 64-bit adder/subtractor.

Fig. 3. Hardware implementation scheme of ternary neural network
PROGRAMMING MODELS AND APPLICATIONS
DEEP LEARNING

• Training
  • Compute: very demanding
  • Network: asynchronous, associative, commutative → Not demanding
  • Precision requirement: single or even less

• Inference
  • Compute: not demanding quantisable
  • Embarrassingly parallel → Not demanding
  • Precision requirement: quantisable up to 1 bit with decent performances
**Facts**

- INFRA-P call Nov. 2017
- Ranked 1st on ~30 submitted projects
- Kick-off mid apr 2018
- 4.5M€ funding
- 2 partners
- 8 associated partners
- Coord. M. Aldinucci
- Many industrial stakeholders
<table>
<thead>
<tr>
<th>Users</th>
<th>Kind of service</th>
<th>Services</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain experts with no skills</strong> on ML and BDA.</td>
<td>Service-as-a-Service (SaaS)</td>
<td>SaaS for ML and BDA designed within HPC4AI partners</td>
<td>Market place for ML and BDA services: Dashboards, trained models in several domains (NLP, Vision, …)</td>
</tr>
<tr>
<td><strong>Domain experts skilled</strong> on ML and BDA. <strong>Not expert in parallel computing.</strong></td>
<td>Platform-as-a-Service (PaaS)</td>
<td>PaaS solutions for ML and BDA directly designed within HPC4AI or companion projects</td>
<td>Market place of VMs and Platforms realising software stacks for ML and BDA. Solutions for data ingestion, data lake, etc.</td>
</tr>
<tr>
<td>New networks or pipelines; training set required.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Researchers, cloud engineering, ML and BDA framework designers, cloud engineers, stack and automation designers.</td>
<td>1) <strong>Infrastructure-as-a-Service (IaaS)</strong></td>
<td>1) GARR/other cloud able to support federation</td>
<td>1) Openstack, docker, VM, object storage, file storage, kubernetes, etc.</td>
</tr>
<tr>
<td></td>
<td>2) <strong>Metal-as-a-Service (IaaS)</strong></td>
<td>2) Job scheduler for HPC resources</td>
<td>2) Alternative cloud, job queue, Big Data Stack (Spark, …).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Researchers, run-time designers.</td>
<td><strong>Hardware</strong></td>
<td>Bare Metal</td>
<td>Multicore, GPU, storage, network, switch, UPS, cooling, etc.</td>
</tr>
</tbody>
</table>
Realtime Syslog Analytics #22

By bigdata-charmers • xenial • Stable

This is a six unit big data cluster that includes Hadoop 2.7.3 and other components from Apache Bigtop. By leveraging Rsyslog and Apache Flume, this bundle provides an environment for analysing syslog events in Apache Zeppelin web notebooks.
EXAMPLE PaaS ON BARE METAL: KUBERNETES-AS-A-SERVICE

EXAMPLE PaaS ON VMs: BIGDATA-AS-A-SERVICE

**Follow-up Project (2018_ICT-11-A EU IA, 13M€)**

**DeepHealth: Deep-Learning and HPC to Boost Biomedical Applications for Health**

- **18 Partners:** Everis, Siveco, Wings, Philips, SIVECO, IBM, Thales, CEA, Treelogic, EPFL, UPV, UNITO, UNIMORE, ...

- **Design and develop:**
  - European Library for Distributed Deep Learning for health
  - AI-on-demand cloud platforms (HPC4AI, ...)
FROM HPC TO BIGDATA TO DEEP LEARNING

- **HPC**
  - Send/recv, batch, CPU intensive
  - Programmer should have the direct knowledge of all processes and all communications

- **Exascale HPC**
  - Data movements rather than compute power
  - Beyond locally synchronous data parallelism? Tasks?

- **BigData (more abstract)**
  - Mostly streams & I/O: intensive, interactive
  - Virtualised, cloud stack, Platform/Service-as-a-Service, service composition
  - O(n) algorithms - no more!

- **Deep Learning (much more abstract)**
  - Naturally asynchronous, permissive, low precision
  - Both batch (training) and stream (inference)
PROBLEMS SOLVED (PROGRAMMING MODEL)

• HPC: large-scale locally synchronous (stencil)
  • Low-level: MPI
  • High-level: open problem. Tasks?
• Big Data analytics
  • MapReduce programming model (data flow run-time)
• Deep learning
  • Training: API and architecture for **scale-up** well understood (GPU/Tensor)
  • Inference: Quantisation + Processing-in-Memory new unfolding

Open Problems (Programming Model)

- DL: Distributed training at scale
  - Today: lock-step all-to-all weight exchange, i.e. globally synchronous → not scalable

OPEN PROBLEMS

• General methodology
  • Most of the paper “observe” network x on data y with tuning z

• Distributed memory hierarchy
  • Data movement prevails computation

• Privacy and data marketplace
  • Multi-party computation
  • Federated learning
  • Inference: a complete programming ecosystem
FUTURE HORIZONS (PROGRAMMING MODEL)

• Specialisation: accelerators are the key
  • Deep Learning
  • Quantum computing
  • Neuromorphic

• Programming models: methodology is the key
  • Should be simple and compositional as sequential or web services coding
  • Performance is not the only issue: time-to-market, cost, correctness, …
FUTURE HORIZONS