FPGAs in Economics

Jesús Fernández-Villaverde,[1] and Alessandro Peri[2]

March 2, 2022

[1]University of Pennsylvania

[2]CU Boulde

Motivation

An idea

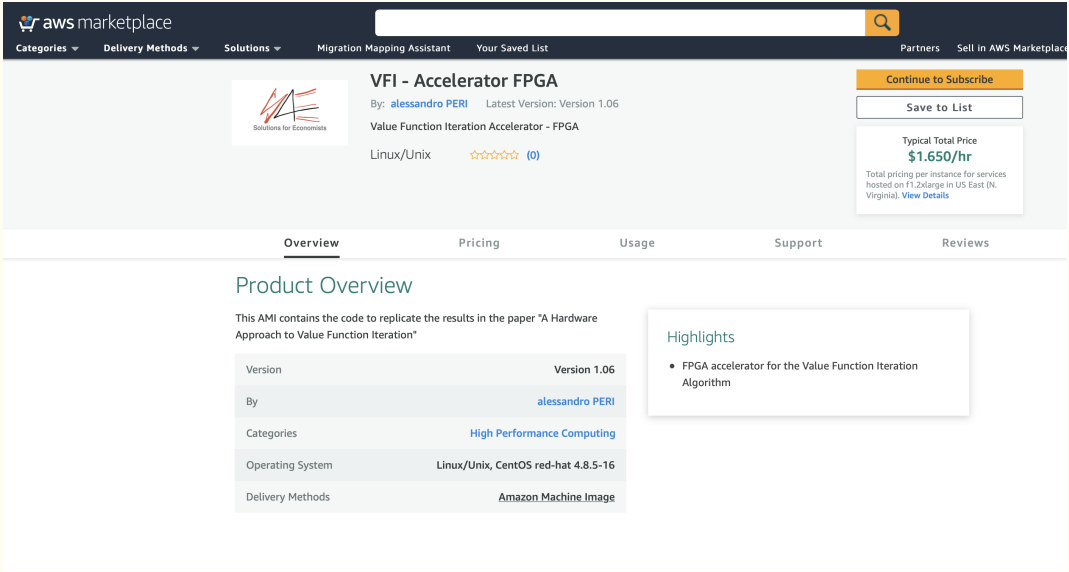
Alan Kay

People who are really serious about software should make their own hardware.

Contribution

- Slides based on A Hardware Approach to Value Function Iteration by Alessandro Peri (with help from Alessandro Nale) and our joint ongoing work.
- Design a chip (FPGA) *specialized* in the solution of RBC model.
- Document $\sim$10x speed gains vis-á-vis GPU data-parallelization.
- **Assembly line parallelism**: instruction and pipeline parallelism.
- The FPGA chip: customizable and easily designed.
- The **FPGA image** built by Alessandro is available in Amazon AWS: `https://aws.amazon.com/marketplace/pp/B07PLWCNCV`

VFI - accelerator FPGA



Hardware solution

- To cope with increasing computational challenges, corporations (like Google, Intel):
  1. have stopped relying on commodity general purpose hardware (like CPUs) and
  2. have started developing their own chips.
- Example: in 2017, Google presented its custom accelerator for machine learning applications: the Tensor Processing Unit (TPU).

Norm Jouppi, Distinguished Hardware Engineer, Google

This is roughly equivalent to fast-forwarding technology about seven years into the future (three generations of Moore's Law).

The FPGA: Field-programmable gate array

The FPGA: Field-programmable gate array

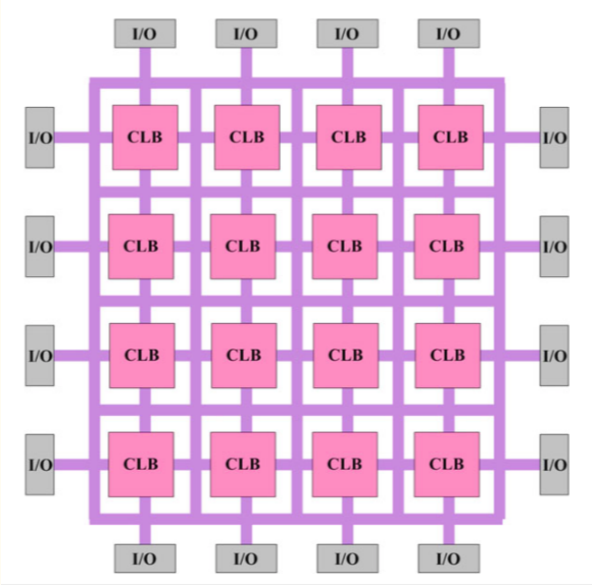
CPU

GPU Graphic Processing Unit

The FPGA: Field-programmable gate array

- FPGA: Field-programmable gate array.
  - Fully customizable (*Design* language).
  - Easily programmable.
  - Not as fast as CPUs or GPUs (250MHz).
- Where do the gain come from? Specialization by pipeline parallelism at the logical resources level.

The FPGA: Field-programmable gate array



The FPGA: Field-programmable gate array

- The FPGA hardware architecture comprises:
  1. Configurable logic blocks (CLBs), i.e. the logical blocks used to implement the user-defined functions.
  2. The network of routing that connects the CLB together.
  3. I/O blocks that allows for off-chips connections, i.e., for the CPU to speak with the FPGA.

An application

The Bellman Equation

$$V(\underline{s}) = \max_{k' \in \Gamma(\underline{s})} F(\underline{s}, k') + \beta \cdot \int_{\underline{s}'} V(\underline{s}') Q(\mathrm{d}s', s)$$

- *Solution*: Value function iteration (VFI):

$$V^i(\underline{s}) = \mathbf{T}^i V^0(\underline{s})$$

- **Problem:** Computationally Expensive
- **Software Solutions**: Alternative Algorithms, Peak-finding, Good Guesses...
- **Hardware Solutions:** Data-parallelization:
  1. MPI-CPU
  2. GPU (Aldrich et al., 2001)

The RBC model

$$V(k,z) = \max_{k'} \frac{(zk^{\alpha} + (1-\delta)k - k')^{1-\eta}}{1-\eta} + \beta \cdot \int_{z'} V(k',z')Q(z',z')\mathrm{d}z'$$

$$\ln z_{t+1} = \rho \ln z_t + \epsilon_{t+1}, \qquad \epsilon_{t+1} \sim N\left(0, \sigma^2\right)$$

Parameter	Value	Description
β	0.984	Subjective discount factor
η	2.000	Arrow-Pratt relative risk aversion coefficient
α	0.350	RTS parameter
δ	0.010	Depreciation rate
ρ	0.950	Persistence of the AR(1) log-productivity process
σ	0.005	Volatility of the innovation of the AR(1) log-productivity process
N_z	**4**	Grid points of z
N_k	**65536**	Grid points of k

Results

Table 1: Time Comparison in Seconds (250MHz)

	FPGA	GPU
Time		
Initialization	0.25	0.01
Solution	**1.42**	**13.60**
Reading	0.75	0.01
Total	2.42	13.62
Platforms' Technical Specification		
Max Clock (MHz)	250	875
Cores	-	4992

Hardware specification

Table 2: Platforms' Technical Specification

Platform	ASIC	CPU
FPGA	16 nm Xilinx UltraScale Plus	Intel(R) Xeon(R) CPU E5-2686 v4 2.30GHz
GPU	NVIDIA Tesla K80	Intel Xeon CPU E5-2680 v3 2.50GHz

Determinants of the speed gains

The Assembly line parallelism

- Two Novel Layers of Parallelism:
  - Instruction Parallelism
  - Pipeline Parallelism
- . . . at the *logical resources level*

The value function iteration algorithm

- Each iteration step i

$$V^i(k,z) = \max_{k' \in K} \underbrace{F(k,z,k') + \beta \cdot \int_{z'} V^{i-1}(k',z')Q(z',z')\mathrm{d}z'}_{\text{Objective Function}}$$

The value function iteration algorithm

- Each iteration step i

$$V^i(k,z) = \underbrace{\max_{k' \in K}}_{\text{Peak-Finding}} \underbrace{F(k,z,k') + \beta \cdot \int_{z'} V^{i-1}(k',z') Q(z',z') \mathrm{d}z'}_{\text{Objective Function}}$$

The value function iteration algorithm

- Each iteration step i

$$V^i(k,z) = \underbrace{\max_{k' \in K}}_{\text{Peak-Finding}} \underbrace{F(k,z,k') + \beta \cdot \int_{z'} V^{i-1}(k',z')Q(z',z')\mathrm{d}z'}_{\text{Objective Function}}$$

Assembly line parallelism

- **Instruction level parallelism**
- **Pipeline parallelism**

Objective function

- Objective Function

$$h(k, z, k') = \frac{(w(k, z) - k')^{1-\eta}}{1 - \eta} + \beta \cdot \sum_{z'} V(k', z') Q(z', z) \mathrm{d}z'$$

Table 3: Arithmetic Operations in the Objective Function

	Adds/ Subs	**Multiplicators**	**Logarithm**	**Exponential**
$F(k, z, k')$	1	2	1	1
$\beta \cdot \sum_{z'} V(k', z') Q(z', z)$	3	4	-	-
$h(k, z, k')$	**5**	**6**	**1**	**1**

Note: Arithmetic operations involved in the evaluation of the objective function

Objective function

- Objective Function

$$h(k, z, k') = \frac{(w(k, z) - k')^{1-\eta}}{1 - \eta} + \beta \cdot \sum_{z'} V(k', z') Q(z', z) \mathrm{d}z'$$

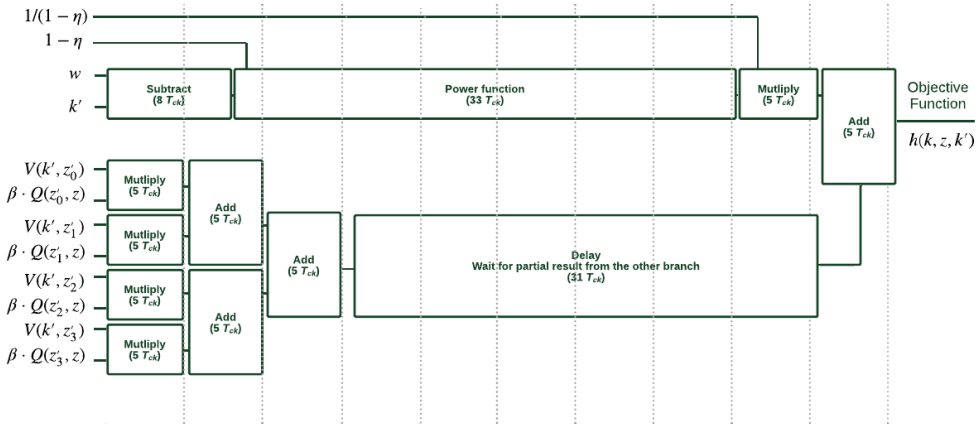
Table 3: Arithmetic Operations in the Objective Function

	Adds/ Subs	**Multiplicators**	**Logarithm**	**Exponential**
$F(k, z, k')$	1	2	1	1
$\beta \cdot \sum_{z'} V(k', z') Q(z', z)$	3	4	-	-
$h(k, z, k')$	**5**	**6**	**1**	**1**

Note: Arithmetic operations involved in the evaluation of the objective function

Objective function

$$h(k,z,k') = \frac{(w(k,z) - k')^{1-\eta}}{1-\eta} + \beta \cdot \sum_{z'} V(k',z')Q(z',z)\mathrm{d}z'$$



The peak-finding algorithm

$$V^i(k,z) = \underbrace{\max_{k' \in K}}_{\text{Peak-Finding}} \quad F(k,z,k') + \beta \cdot \int_{z'} V^{i-1}(k',z')Q(z',z')\mathrm{d}z'$$

- 3 k'-Index Evaluation Binary search algorithm with lookup table.

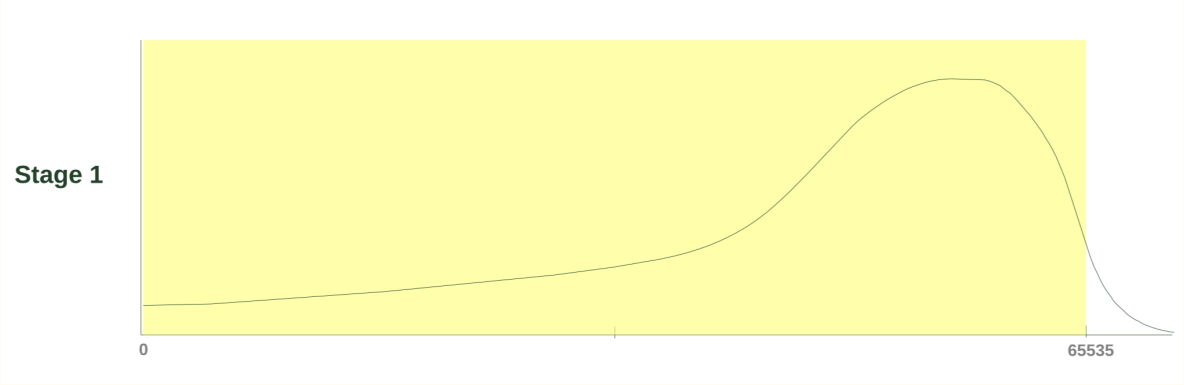
The peak-finding algorithm

$$V^i(k,z) = \underbrace{\max_{k' \in K}}_{\text{Peak-Finding}} \quad F(k,z,k') + \beta \cdot \int_{z'} V^{i-1}(k',z')Q(z',z')\mathrm{d}z'$$

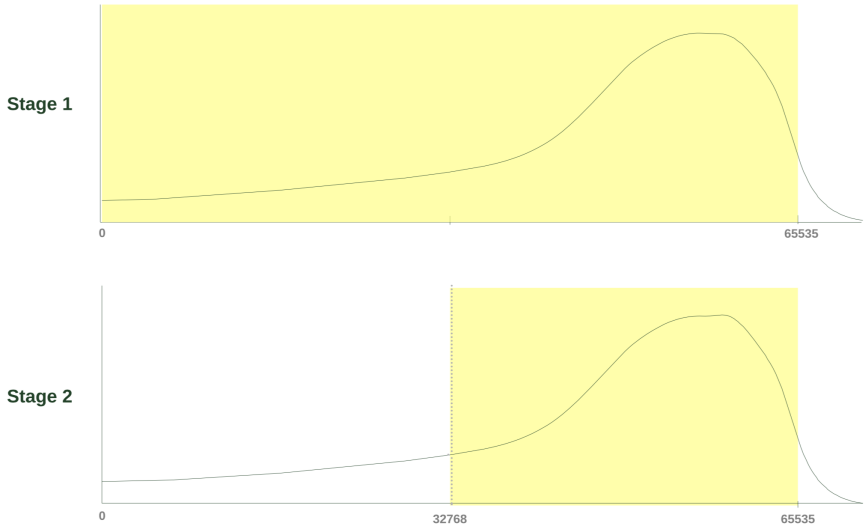
- 3 k'-Index Evaluation: Binary search algorithm with lookup table.

Binary search algorithm

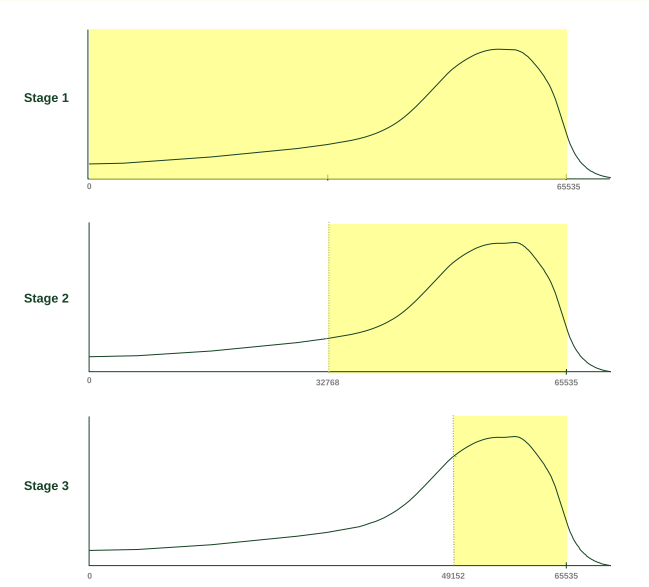
The binary search algorithm



The binary search algorithm



The binary search algorithm



The peak-finding algorithm

$$V^i(k,z) = \underbrace{\max_{k' \in K}}_{\text{Peak-Finding}} \quad F(k,z,k') + \beta \cdot \int_{z'} V^{i-1}(k',z') Q(z',z') \mathrm{d}z'$$

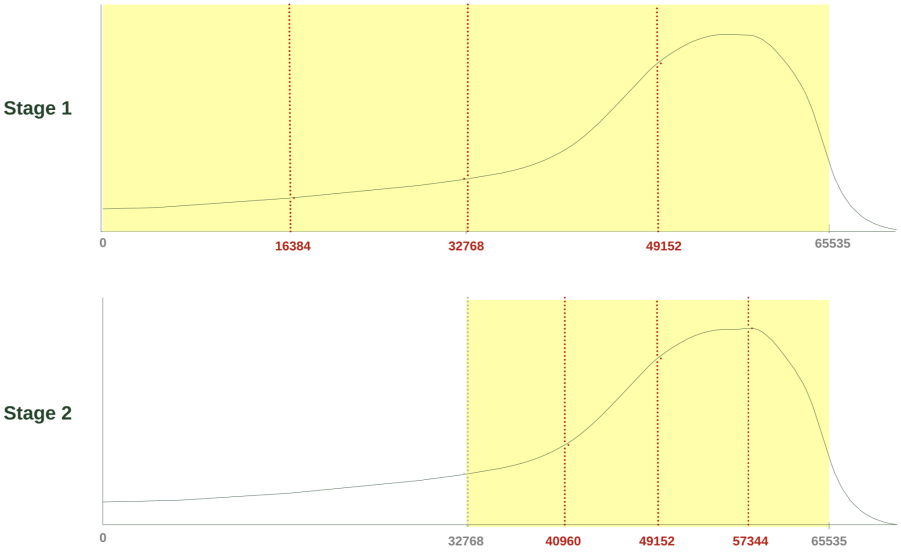
- 3 k'-Index Evaluation binary search algorithm with lookup table.
  - Solution in 16 stages ($\log_2 N_k = \log_2 65536 = 16$)

The peak-finding algorithm

$$V^i(k,z) = \underbrace{\max_{k' \in K}}_{\text{Peak-Finding}} \quad F(k,z,k') + \beta \cdot \int_{z'} V^{i-1}(k',z')Q(z',z')\mathrm{d}z'$$

- 3 k'-index evaluation: binary search algorithm with lookup table.
  - Solution in 16 stages ($\log_2 N_k = \log_2 65536 = 16$).
  - Evaluate Objective Function at $\{k'_1, k'_2, k'_3\}$ . . . in parallel.

The binary search algorithm



The peak-finding algorithm

$$V^i(k,z) = \underbrace{\max_{k' \in K}}_{\text{Peak-Finding}} \quad F(k,z,k') + \beta \cdot \int_{z'} V^{i-1}(k',z')Q(z',z')\mathrm{d}z'$$

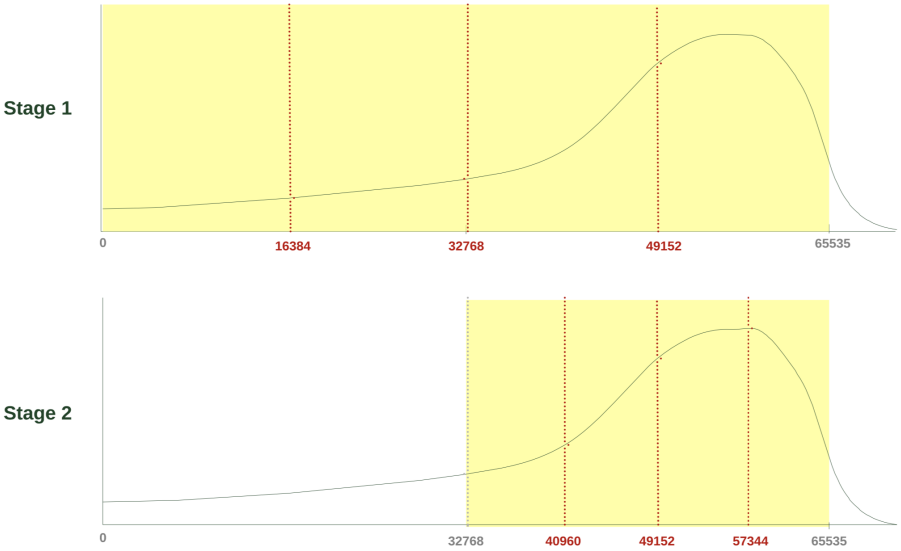
- 3 k'-index evaluation: Binary search algorithm with lookup table
  - Solution in 15 stages ($\log_2 N_k - 1$), last stage 4 indexes)
  - Evaluate objective function at $\{k_1', k_2', k_3'\}$ ...in parallel

The peak-finding algorithm

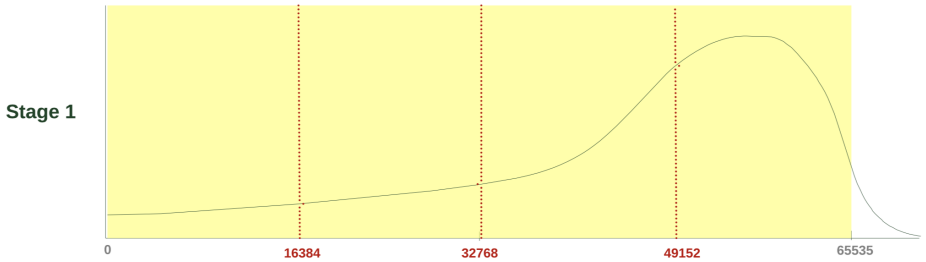
$$V^i(k,z) = \underbrace{\max_{k' \in K}}_{\text{Peak-Finding}} \quad F(k,z,k') + \beta \cdot \int_{z'} V^{i-1}(k',z')Q(z',z')\mathrm{d}z'$$

- 3 k'-index evaluation: Binary search algorithm with lookup table
  - Solution in 15 stages ($\log_2 N_k - 1$), last stage 4 indexes)
  - Evaluate objective function at $\{k'_1, k'_2, k'_3\}$ ... in parallel

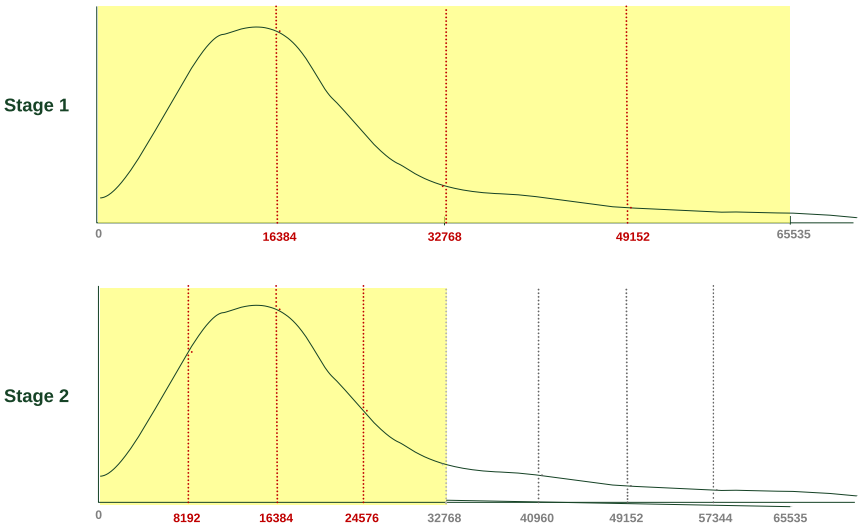
The binary search algorithm



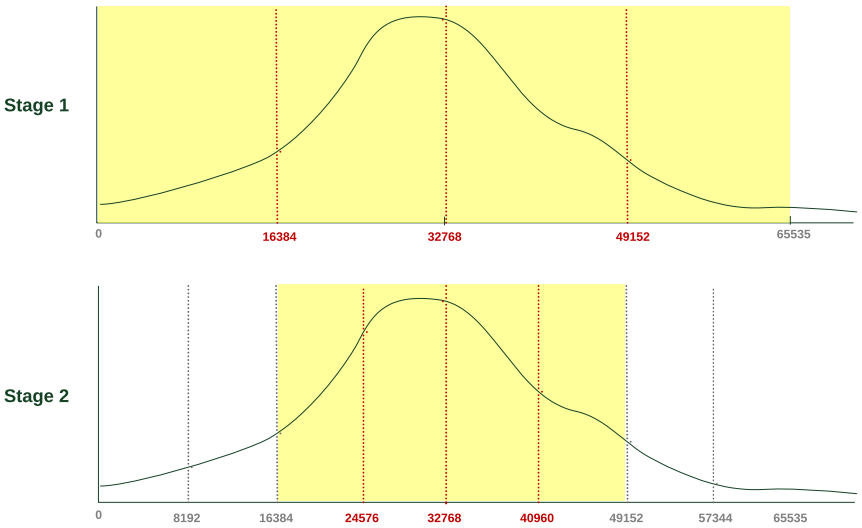
The binary search algorithm



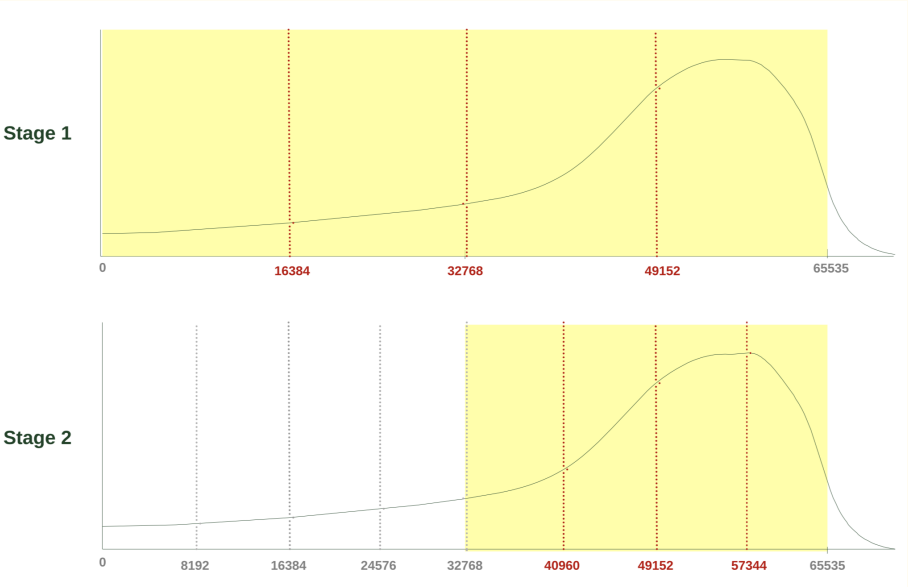
The binary search algorithm



The binary search algorithm



The binary search algorithm



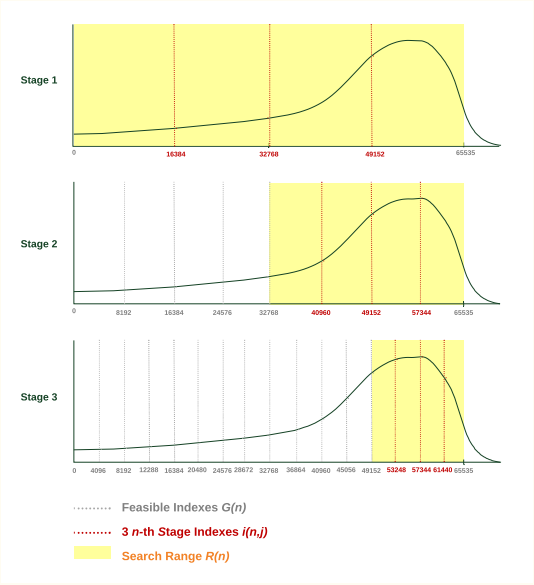
The binary search algorithm

$$j^*(n) = \begin{cases} 0 & \text{if} \quad h(j^*=1) \quad \text{is the max} \\ 1 & \text{if} \quad h(j^*=1) = h(j^*=2) \quad \text{are the max} \\ 2 & \text{if} \quad h(j^*=2) \quad \text{is the max} \\ 2 & \text{if} \quad h(j^*=1) = h(j^*=2) = h(j^*=3) \\ 3 & \text{if} \quad h(j^*=2) = h(j^*=3) \quad \text{are the max} \\ 4 & \text{if} \quad h(j^*=3) \quad \text{is the max} \end{cases}$$

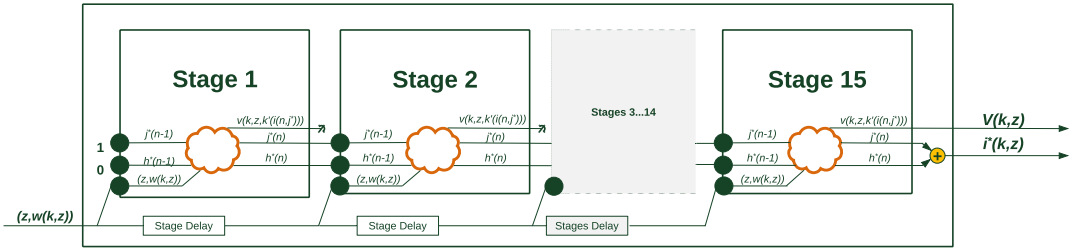
$$h^*(n+1) = 2 \cdot h^*(n) + j^*(n)$$

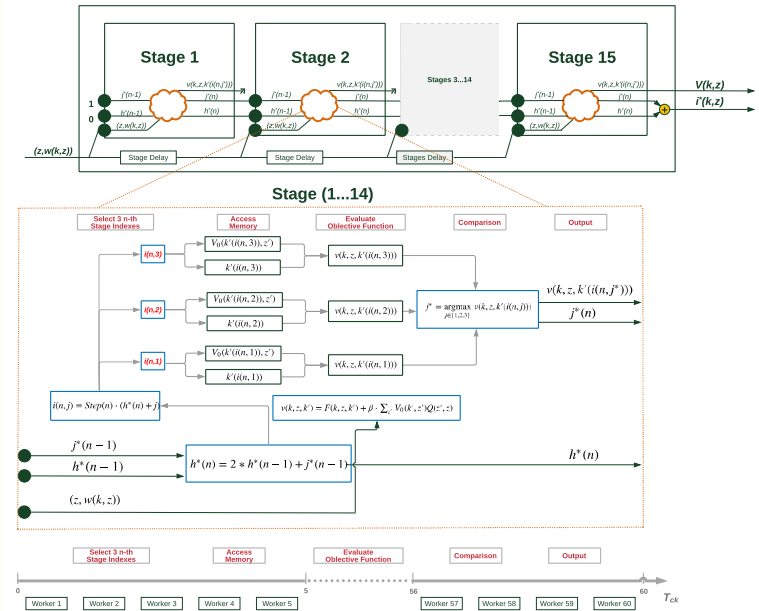
$$i(n,j) = \underbrace{\frac{N_k}{2^{n+1}}}_{Step(n)} \cdot (h^*(n) + j) \qquad j = 1,2,3$$

The binary search algorithm

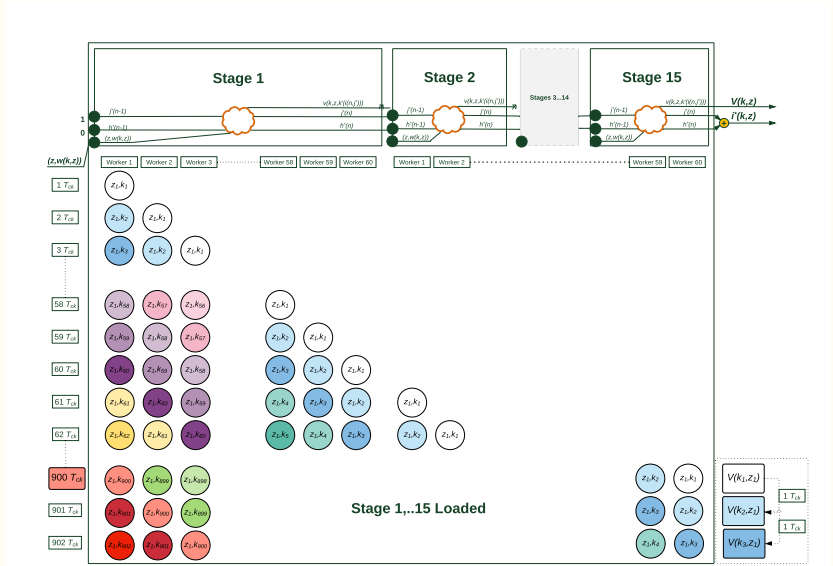


Pipe-line parallelism




Stage 1
Stage 2
Stages 3...14
Stage 15
V(k,z)
i*(k,z)
(z,w(k,z))
Stage Delay
Stage Delay
Stages Delay
Stage (1...14)
Select 3 n-th Stage Indexes
Access Memory
Evaluate Oblective Function
Comparison
Output
i(n,3)
i(n,2)
i(n,1)
j*(n)
h*(n)
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5
Worker 57
Worker 58
Worker 59
Worker 60
Tck

The assembly line parallelism workflow



Solution time decomposition

$$\text{Time} = \underbrace{1352}_{\text{No Iterations}} \cdot \underbrace{65536}_{N_k} \cdot \underbrace{4}_{N_z} \cdot \underbrace{\frac{1}{250 \cdot 1e6}}_{\text{Clock}} = 1.42\, sec$$

Potential applications

- HA models.
- Machine learning.
- Montecarlo simulation.
- Structural estimation.

Costs

- **FPGA** 16 nm Xilinx UltraScale Plus: $\sim$7000$ (alternative, Amazon AWS).
- **Coding time** VHDL (high costs of entry). **Solution:**
  - Collaboration.
  - *FPGA compilers*.
- **Institutional costs?**