Granularity versus Parallelization Overhead in High-Performance Computing Systems

Dongya Koh

University of Arkansas

October 9-10, 2018

Motivation

- Solving a dynamic model with various kinds of heterogeneity needs to rely on numerical methods.
- Over the decades, a number of alternative solution methods were developed to improve the computation time and accuracy.
- The advent of multicore processors has accelerated the use of parallel computation.
- To further speed up, high-performance computing (HPC) system is now available.

A Basic Question:

- To speed up the computational time, how can we take advantage of HPC?

How can we take advantage of HPC?

The right choice of

1. Programming languages
   C/C++, Fortran, Matlab, Python, Julia, R, etc.
2. Solution methods
   VFI, PFI, Projection, Perturbation, EGM, etc.
3. Scalability (how effectively parallelizable)
4. Granularity (the amount of work in the parallel task)
5. Number of processing elements/cores

Programming Languages

Language	Type	Rel. Exec. Time
C/C++	low-level, fastest with gcc	1.00
Fortran	low-level	1.05
Python	high-level, open-source, growing in popularity	$44\times \sim 270\times$
Julia	high-level, new open-source	$2.64\times \sim 2.70\times$
R	high-level, open-source	$281\times \sim 475\times$
Matlab	high-level, not free, license issue	$9\times \sim 11\times$
Mathematica	high-level	$809\times$

Source: Arouba & Fernandez-Villaverde (2014)

Solution Methods

Low Dimension	High Dimension
Value function iteration (VFI)	Smolyak sparse grid method (Krueger & Kubler, 2004, etc.)
Policy function iteration (PFI)	Adaptive grid method (Brumm & Scheidegger, 2017)
Projection method (Judd, 1992, etc.)	Stochastic simulation algorithm (Den Haan & Marcet, 1990, etc.)
Endogenous grid method (EGM) (Carroll, 2005, etc.)	ε-distinguishable set method (Judd, Maliar, Maliar, 2015)
Envelope condition method (Maliar & Maliar, 2013)	Cluster grid method (Judd, Maliar, Maliar, 2015)
Precomputation method (Judd, Maliar, Maliar, & Tsener, 2017)	Perturbation method (Judd & Guu, 1993, etc.)

Optimal Number of (Physical or Virtual) Cores

- The total runtime of a program with n cores:

$$T(n) = \frac{T_p}{n} + T_s + P(n)$$

- The optimal number of cores to minimize the runtime:

$$P'(n^*) = \frac{T_p}{n^{*2}}$$

- If the parallelization overhead is approximated as $P(n) = pn^\alpha \quad (p, \alpha > 0)$:

$$n^* = \left(\frac{T_p}{\alpha p}\right)^{1+\alpha}$$

Parallelization Overhead

- Sources of parallelization overhead:
  1. Communication overhead in the form of synchronization and data communications
  2. Idling due to load imbalances
- Overheads vary by the implemented parallel algorithms and problems to be solved
- Parallelization overhead with n cores can be calculated as

$$P(n) = (T(n) - T_s) - \frac{T_p}{n},$$

provided that $P(1) = 0$

Overview

What We Do:

- Solve a variety of life-cycle models with efficient/inefficient solution methods.
- Compute the parallelization overhead.
- Compare the runtime of each parallel program by granularity and the number of cores.

What We Find:

- Parallelization overhead sets an upper bound to the HPC speed-ups.
- A design of HPC-efficient algorithm to reduce the parallelization overhead is essential for speed-ups in HPC.

HPC Environment

HPC System:

- HPC consists of three sub clusters
- Interconnected with a 324-port QDR 40 Gbps nonblocking QLogic Infiniband switch and supplementary switches
- 88TB long-term storage and 35 TB of scratch storage
- OS: Centos 6.5

Specifications of General Use Queue:

- CPU: 4x AMD Opteron 16-core 2.3 GHz 6276
- Memory/node: 512GB
- Max PBS spec: nodes=2:ppn=64
- Max PBS time: 72:00:00
- Software: Python/3.6.0-Anaconda

Model 1: Consumption/Saving

- The baseline model assumes that an individual lives until age T.
- The individual's problem is to choose an amount of saving and consumption

$$V_t(a_t, e_t) = \max_{c_t, a_{t+1}} u(c_t) + \beta \mathbb{E}_t V_{t+1}(a_{t+1}, e_{t+1})$$

$$\begin{aligned} \text{s.t.} \quad c_t + a_{t+1} &= we_t + (1+r)a_t \\ a_{t+1} &\geq \underline{a} \\ e_{t+1} &\sim P(e_{t+1}|e_t). \end{aligned}$$

- The utility function takes isoelastic preference, $u(c) = \frac{c^{1-\sigma}}{1-\sigma}$

Model 1: VFI Algorithm

Step 1. Initialization

a. Set model parameters.
b. Define grid points for $(a_t, e_t) \in \mathcal{A} \otimes \mathcal{E}$.
c. Construct a transition matrix $P(e_{t+1}|e_t)$.

Step 2. Computing a household problem at $t = T$

a. For any given (a_T, e_T), $a_{T+1} = 0$. Therefore, we obtain

$$c_T = we_T + (1 + r)a_T$$
$$V_T(a_T, e_T) = u(c_T)$$

Step 3. Computing a household problem at $t < T$

a. For each point (a_t, e_t) with V_{t+1}, compute for all a_{t+1},

$$W_t(a_t, e_t) = u(we_t + (1 + r)a_t - a_{t+1}) + \beta \mathbb{E}_t V_{t+1}(a_{t+1}, e_{t+1})$$

b. Then, choose

$$\max_{a_{t+1}} W_t(a_t, e_t) = V_t(a_t, e_t).$$

Model 1: VFI Algorithm

Step 1. Initialization

a. Set model parameters.
b. Define grid points for $(a_t, e_t) \in \mathcal{A} \otimes \mathcal{E}$.
c. Construct a transition matrix $P(e_{t+1}|e_t)$.

Step 2. Computing a household problem at $t = T$

a. For any given (a_T, e_T), $a_{T+1} = 0$. Therefore, we obtain

$$c_T = we_T + (1 + r)a_T$$
$$V_T(a_T, e_T) = u(c_T)$$

Step 3. Computing a household problem at $t < T$

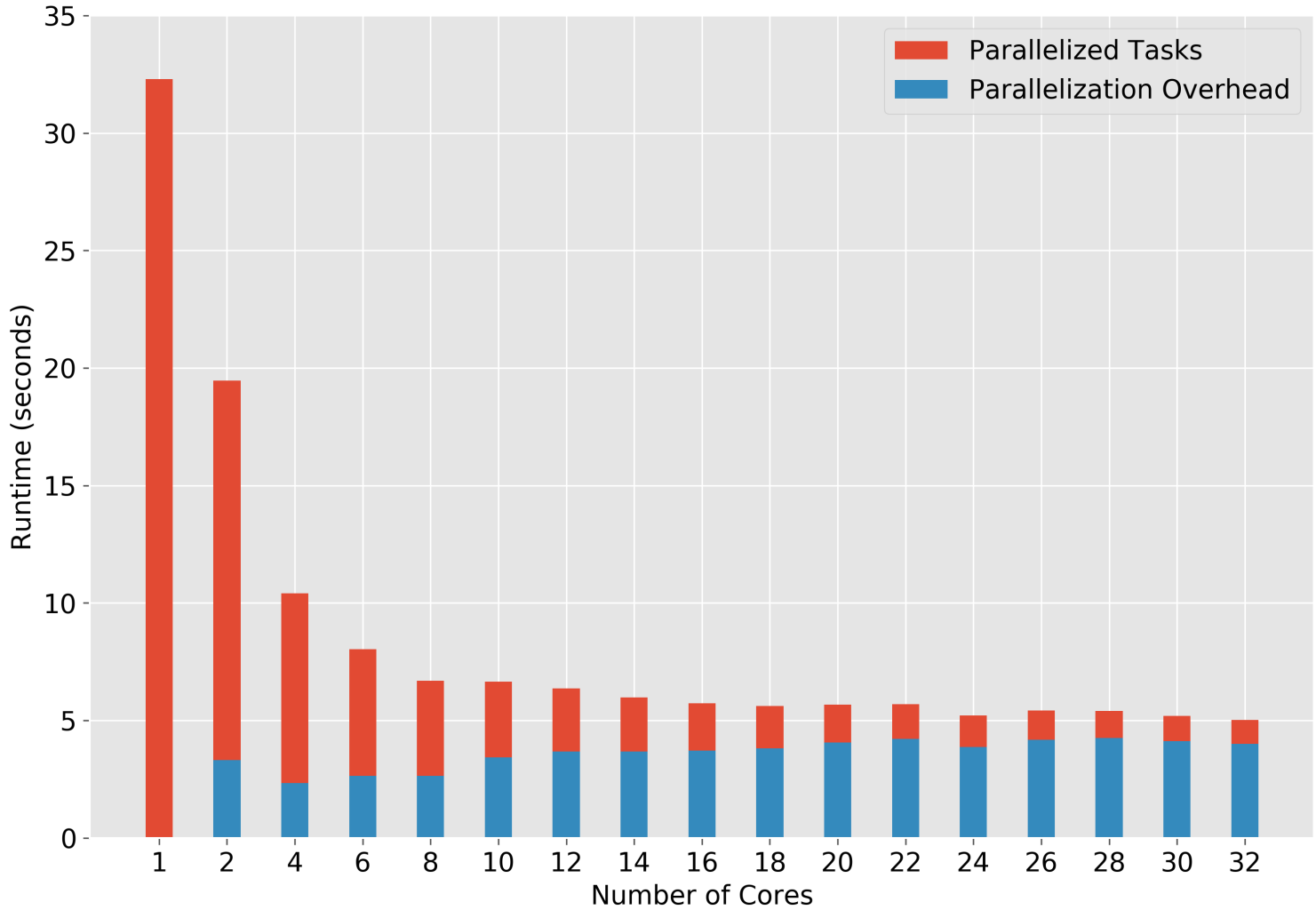
a. For each point (a_t, e_t) with V_{t+1}, compute for all a_{t+1},

$$W_t(a_t, e_t) = u(we_t + (1 + r)a_t - a_{t+1}) + \beta \mathbb{E}_t V_{t+1}(a_{t+1}, e_{t+1})$$

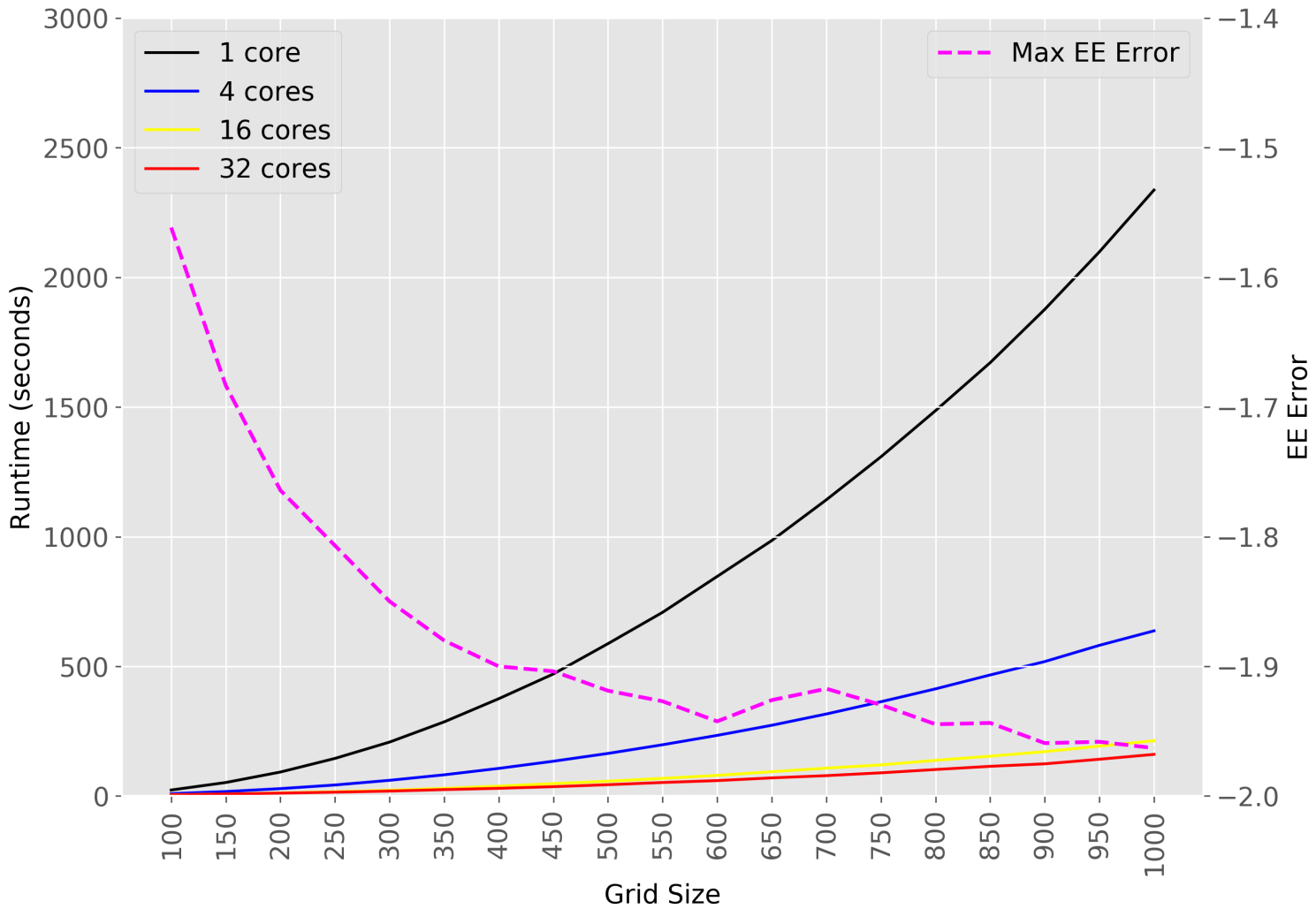
b. Then, choose

$$\max_{a_{t+1}} W_t(a_t, e_t) = V_t(a_t, e_t).$$

Model 1: VFI Running Time by Cores



Model 1: VFI Running Time and Accuracy by Grids



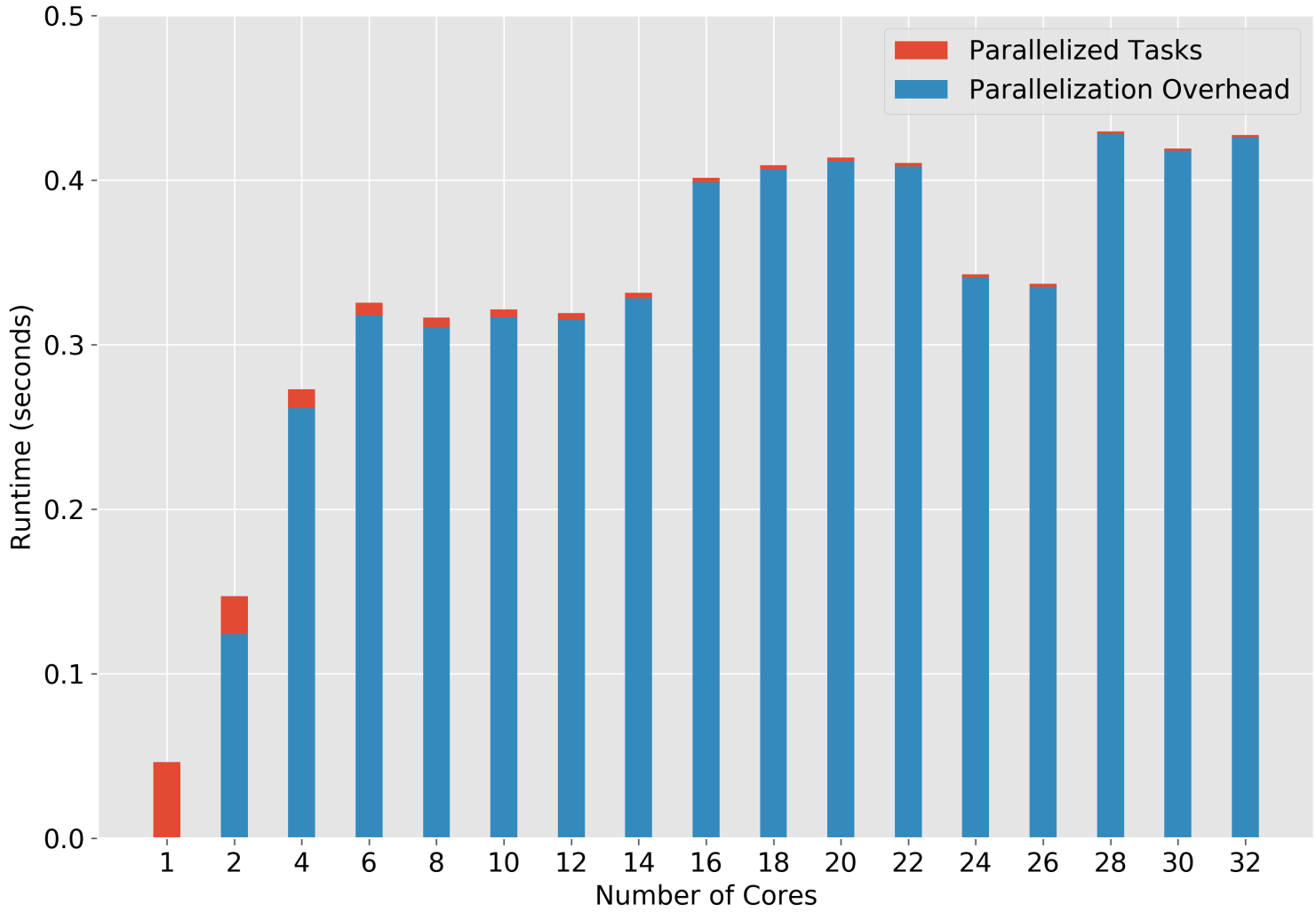
Model 1: EGM Algorithm

Step 1.	Initialization
	...
Step 2.	(EXGM) Computing a household problem at $t = T$
a.	For any given (a_T, e_T), $a_{T+1} = 0$. Therefore, we obtain $c_T = we_T + (1+r)a_T$ $V_T^a = u'(c_T)(1+r)$
Step 3.	(ENGM/EXGM) Computing a household problem at $t < T$
a.	(FOC) $\beta \mathbb{E} V_{t+1}^a = u'(c_t)$: compute $\hat{c}_t$ for each point in (a_{t+1}, e_t).
b.	(BC) $c_t = we_t + (1+r)a_t - a_{t+1}$: compute $\hat{a}_t$ for each $(\hat{c}_t, a_{t+1}, e_t)$.
c.	Given the obtained policy functions $g_t^c(\hat{a}_t, e_t) = \hat{c}_t$, $g_{t+1}^a(\hat{a}_t, e_t) = a_{t+1}$, interpolate new policy functions for each grid point (a_t, e_t).
d.	(EC) $V_t^a = u'(c_t)(1+r)$, we compute V_t^a .

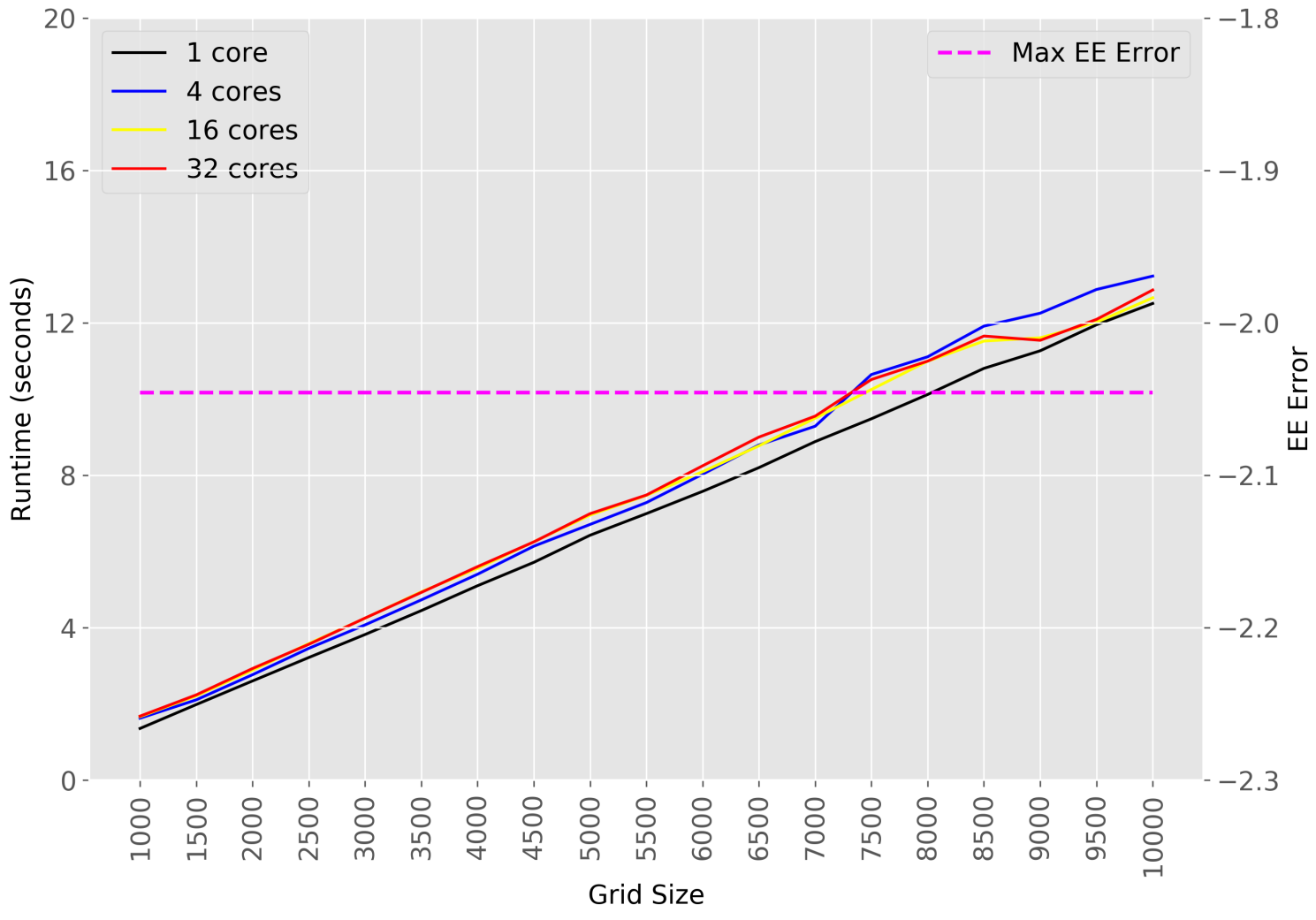
Model 1: EGM Algorithm

Step 1.	Initialization
	...
Step 2.	(EXGM) Computing a household problem at $t = T$
a.	For any given (a_T, e_T), $a_{T+1} = 0$. Therefore, we obtain $c_T = we_T + (1+r)a_T$, $V_T^a = u'(c_T)(1+r)$
Step 3.	(ENGM/EXGM) Computing a household problem at $t < T$
a.	(FOC) $\beta \mathbb{E} V_{t+1}^a = u'(c_t)$: compute $\hat{c}_t$ for each point in (a_{t+1}, e_t).
b.	(BC) $c_t = we_t + (1+r)a_t - a_{t+1}$: compute $\hat{a}_t$ for each $(\hat{c}_t, a_{t+1}, e_t)$.
c.	Given the obtained policy functions $g_t^c(\hat{a}_t, e_t) = \hat{c}_t$, $g_{t+1}^a(\hat{a}_t, e_t) = a_{t+1}$, interpolate new policy functions for each grid point (a_t, e_t).
d.	(EC) $V_t^a = u'(c_t)(1+r)$, we compute V_t^a .

Model 1: EGM Running Time by Cores



Model 1: EGM Running Time and Accuracy by Grids



Model 2: Elastic Labor Supply

- The individual's problem is to choose an amount of saving, consumption, and labor supply.

$$V_t(a_t, e_t) = \max_{c_t, n_t, a_{t+1}} u(c_t, n_t) + \beta \mathbb{E}_t V_{t+1}(a_{t+1}, e_{t+1})$$

$$\begin{aligned} \text{s.t.} \quad c_t + a_{t+1} &= w e_t n_t + (1+r) a_t \\ a_{t+1} &\geq \underline{a} \\ e_{t+1} &\sim P(e_{t+1} | e_t). \end{aligned}$$

- the utility function takes isoelastic preference, $u(c, n) = \frac{c^{1-\sigma}}{1-\sigma} - \chi \frac{n^{1+\eta}}{1+\eta}$.

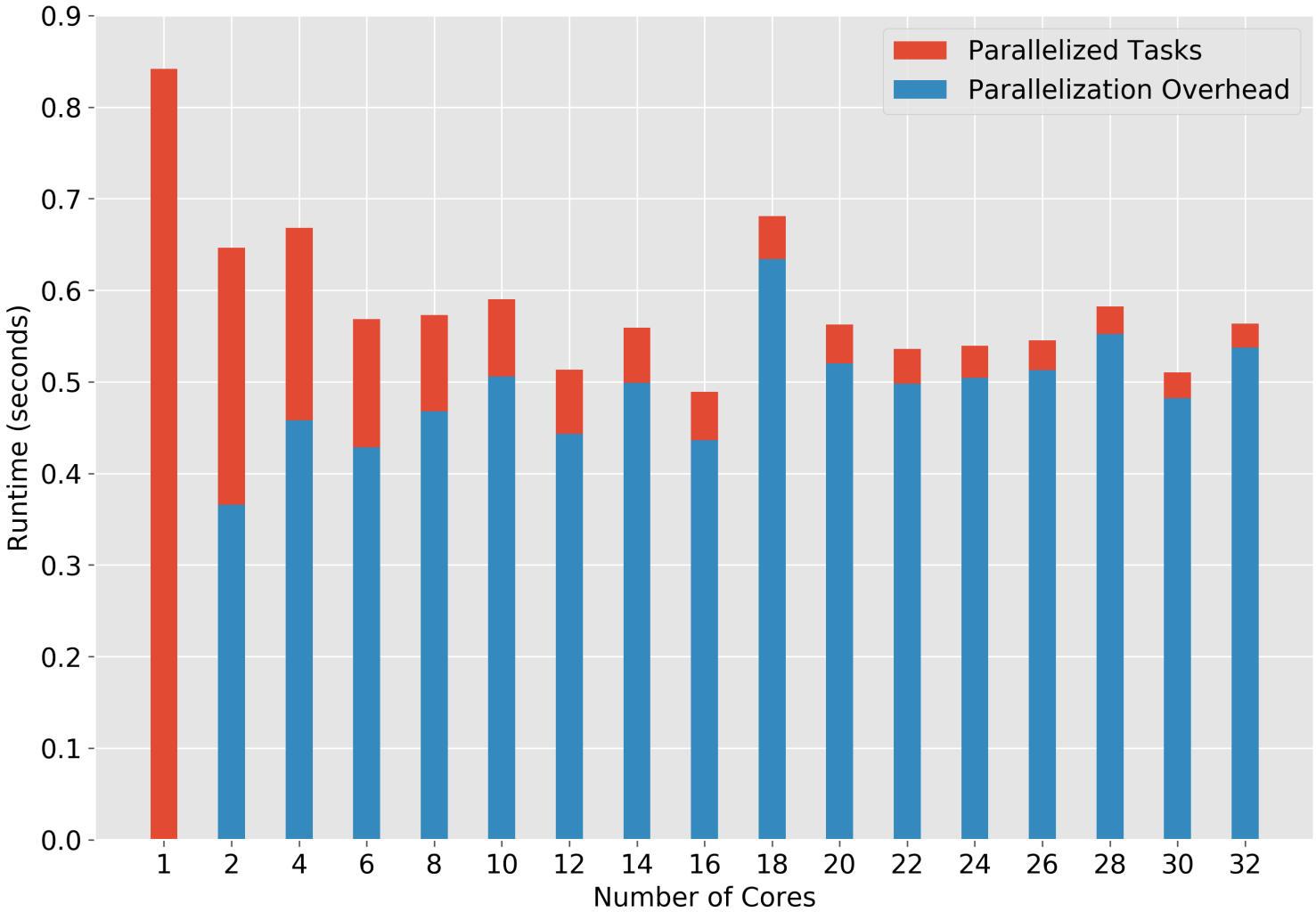
MODEL 2: EGM ALGORITHM

Step 1.	Initialization
	...
Step 2.	**(EXGM) Computing a household problem at $t = T$**
a.	For each point (a_T, e_T), $a_{T+1} = 0$.
b.	(FOC1) $\chi n_T^{\eta} = we_T(we_T n_T + (1+r)a_T)^{-\sigma}$: solve a nonlinear problem for $\hat{n}_T$
c.	(BC) $c_T = we_T n_T + (1+r)a_T$: compute $\hat{c}_T$.
d.	(EC) $V_T^a(a_T, e_T) = u_c(c_T, n_T)(1+r)$: compute V_T^a.
Step 3.	**(ENGM/EXGM) Computing a household problem at $t < T$**
a.	(FOC2) $c_t^{-\sigma} = \beta \mathbb{E} V_{t+1}^a$: compute $\hat{c}_t$ for each point in (a_{t+1}, e_t).
b.	From (FOC1), compute $\hat{n}_t$.
c.	(BC) $a_{t+1} = we_t n_t + (1+r)a_t - c_t$: compute $\hat{a}_t$.
d.	Given the obtained policy functions $g_t^c(\hat{a}_t, e_t) = \hat{c}_t$, $g_t^n(\hat{a}_t, e_t) = \hat{n}_t$, $g_{t+1}^a(\hat{a}_t, e_t) = a_{t+1}$, we interpolate new policy functions, $g_t^c(a_t, e_t) = \tilde{c}_t$, $g_t^n(a_t, e_t) = \tilde{n}_t$, and $g_{t+1}^a(a_t, e_t) = \tilde{a}_{t+1}$ for each grid point $(a_t, e_t) \in \mathcal{A} \otimes \mathcal{E}$.
e.	If $g_{t+1}^a(a_t, e_t) = \tilde{a}_{t+1} < \underline{a}$, then do Step 2 only for the specific $(a_t, e_t) \in \mathcal{A} \otimes \mathcal{E}$. Otherwise, we compute V_t^a from (EC).

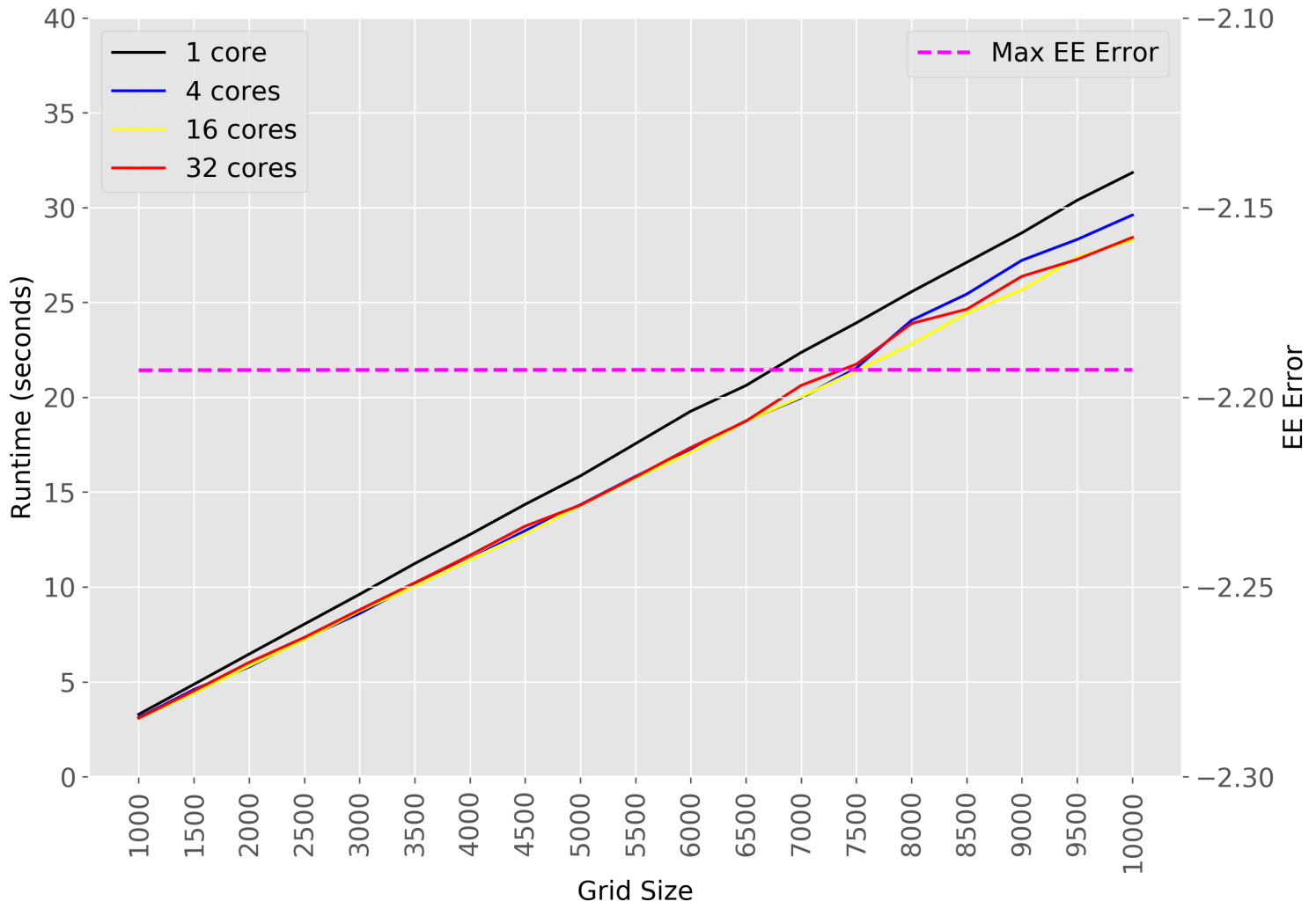
Model 2: EGM Algorithm

Step 1.	Initialization
	...
Step 2.	(EXGM) Computing a household problem at $t = T$
a.	For each point (a_T, e_T), $a_{T+1} = 0$.
b.	(FOC1) $\chi n_T^{\eta} = we_T(we_T n_T + (1+r)a_T)^{-\sigma}$: solve a nonlinear problem for $\hat{n}_T$
c.	(BC) $c_T = we_T n_T + (1+r)a_T$: compute $\hat{c}_T$.
d.	(EC) $V_T^a(a_T, e_T) = u_c(c_T, n_T)(1+r)$: compute V_T^a.
Step 3.	(ENGM/EXGM) Computing a household problem at $t < T$
a.	(FOC2) $c_t^{-\sigma} = \beta \mathbb{E} V_{t+1}^a$: compute $\hat{c}_t$ for each point in (a_{t+1}, e_t).
b.	From (FOC1), compute $\hat{n}_t$.
c.	(BC) $a_{t+1} = we_t n_t + (1+r)a_t - c_t$: compute $\hat{a}_t$.
d.	Given the obtained policy functions $g_t^c(\hat{a}_t, e_t) = \hat{c}_t$, $g_t^n(\hat{a}_t, e_t) = \hat{n}_t$, $g_{t+1}^a(\hat{a}_t, e_t) = a_{t+1}$, we interpolate new policy functions, $g_t^c(a_t, e_t) = \tilde{c}_t$, $g_t^n(a_t, e_t) = \tilde{n}_t$, and $g_{t+1}^a(a_t, e_t) = \tilde{a}_{t+1}$ for each grid point $(a_t, e_t) \in \mathcal{A} \otimes \mathcal{E}$.
e.	If $g_{t+1}^a(a_t, e_t) = \tilde{a}_{t+1} < \underline{a}$, then do Step 2 only for the specific $(a_t, e_t) \in \mathcal{A} \otimes \mathcal{E}$. Otherwise, we compute V_t^a from (EC).

Model 2: EGM Running Time by Cores



Model 2: EGM Running Time and Accuracy by Grids



Model 3: Human Capital Investment

- The individual's problem is to choose an amount of saving, consumption, and human capital investment

$$V_t(a_t, h_t, e_t) = \max_{c_t, a_{t+1}, s_t, h_{t+1}} u(c_t) + \beta \mathbb{E} V_{t+1}(a_{t+1}, h_{t+1}, e_{t+1})$$

$$\begin{aligned} \text{s.t.} \quad c_t + a_{t+1} &= we_t h_t (1 - s_t) + (1 + r) a_t \\ h_{t+1} &= (1 - \delta) h_t + A_h h_t^{\alpha} s_t^{\gamma} \\ s_t &\in [0, 1] \\ a_{t+1} &\geq \underline{a} \\ e_{t+1} &\sim P(e_{t+1} | e_t). \end{aligned}$$

- The utility function takes isoelastic preference, $u(c) = \frac{c^{1-\sigma}}{1-\sigma}$

Model 3: EGM Algorithm

Step 1.	Initialization
	...
Step 2.	**(EXGM) Computing a household problem at $t = T$**
a.	For each point (a_T, h_T, e_T), $a_{T+1} = s_T = 0$.
b.	(BC) $c_T = we_T h_T + (1+r)a_T$: compute $\hat{c}_T$.
c.	(EC1) $V_T^a = u'(c_T)(1+r)$: compute V_T^a.
d.	(EC2) $V_T^h = u'(c_T) we_T \left(1 - \left(1 - \frac{\alpha}{\gamma}\right) s_T + \frac{1-\delta}{\gamma A_h h_T^{\alpha-1} s_T^{\gamma-1}}\right)$: compute V_T^h.

Model 3: EGM Algorithm

Step 3.	(ENGM/EXGM) Computing a household problem at $t < T$

a. (FOC1) $\mathbb{E}V^a_{t+1} we_t h_t = \mathbb{E}V^h_{t+1}\gamma A_h h_t^{\alpha} s_t^{\gamma-1}$: solve a root-finding problem for $\hat{s}_t$ for each point (a_{t+1}, h_t, e_t).

b. (FOC2) $u'(c_t) = \beta \mathbb{E}V^a_{t+1}$: compute $\hat{c}_t$.

c. (BC) $c_t + a_{t+1} = we_t h_t(1 - s_t) + (1 + r)a_t$: compute $\hat{a}_t$.

d. Given the obtained policy functions $g^c_t(\hat{a}_t, h_t, e_t) = \hat{c}_t$, $g^s_t(\hat{a}_t, h_t, e_t) = \hat{s}_t$, $g^a_{t+1}(\hat{a}_t, h_t, e_t) = a_{t+1}$, interpolate new policy functions, $g^c_t(a_t, h_t, e_t) = \tilde{c}_t$, $g^s_t(a_t, h_t, e_t) = \tilde{s}_t$, and $g^a_{t+1}(a_t, h_t, e_t) = \tilde{a}_{t+1}$ for each grid point (a_t, h_t, e_t).

e. If $g^a_{t+1}(a_t, h_t, e_t) = \tilde{a}_{t+1} < \underline{a}$, then do Step 3(a) only for the specific (a_t, h_t, e_t) and $a_{t+1} = \underline{a}$. Otherwise, we compute V^a_t and V^h_t from (EC1) and (EC2).

Model 3: EGM Algorithm

Step 3. (ENGM/EXGM) Computing a household problem at $t < T$

a. (FOC1) $\mathbb{E}V^a_{t+1} we_t h_t = \mathbb{E}V^h_{t+1} \gamma A_h h_t^{\alpha} s_t^{\gamma-1}$: solve a root-finding problem for $\hat{s}_t$ for each point (a_{t+1}, h_t, e_t).

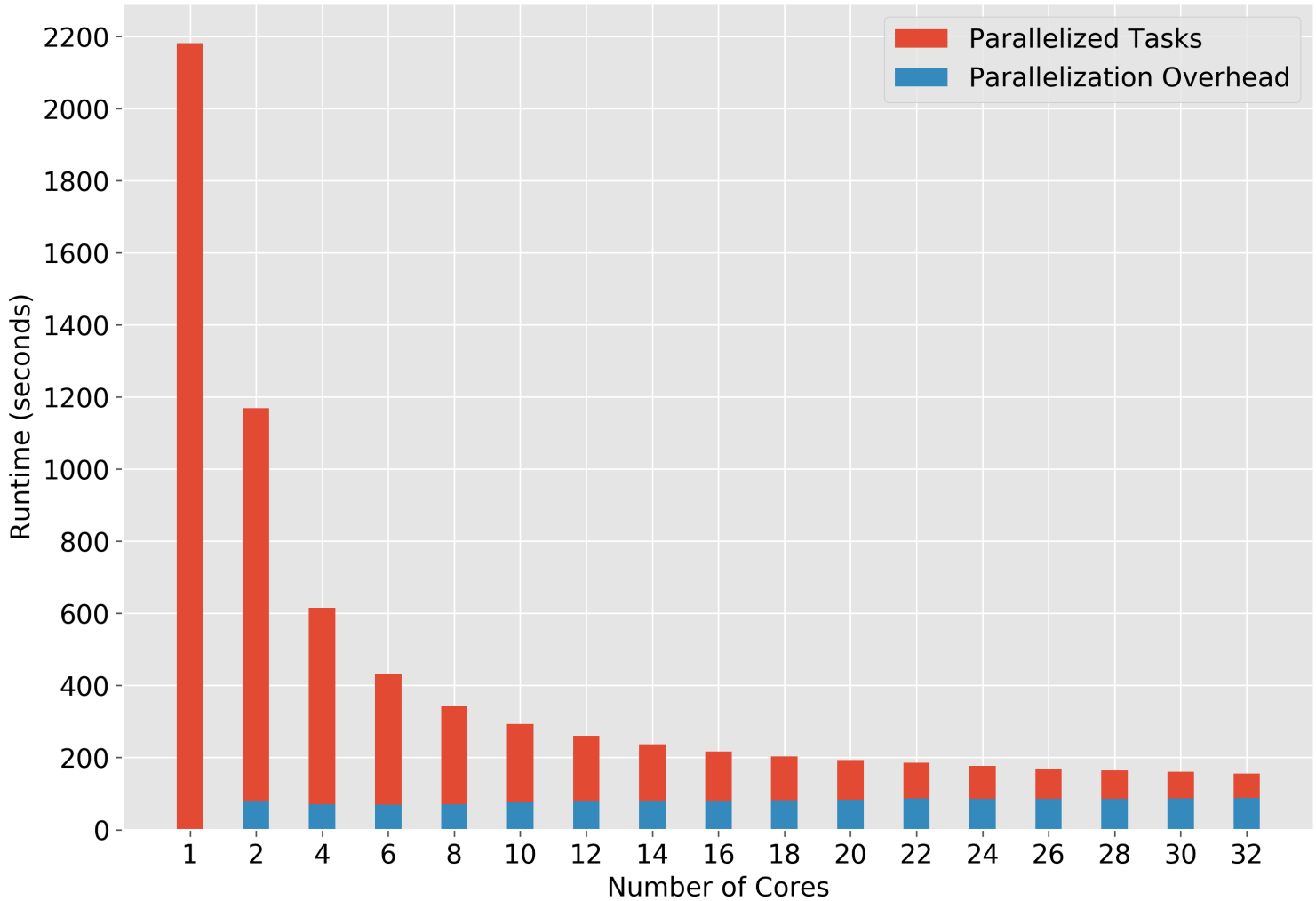
b. (FOC2) $u'(c_t) = \beta \mathbb{E}V^a_{t+1}$: compute $\hat{c}_t$.

c. (BC) $c_t + a_{t+1} = we_t h_t(1 - s_t) + (1 + r)a_t$: compute $\hat{a}_t$.

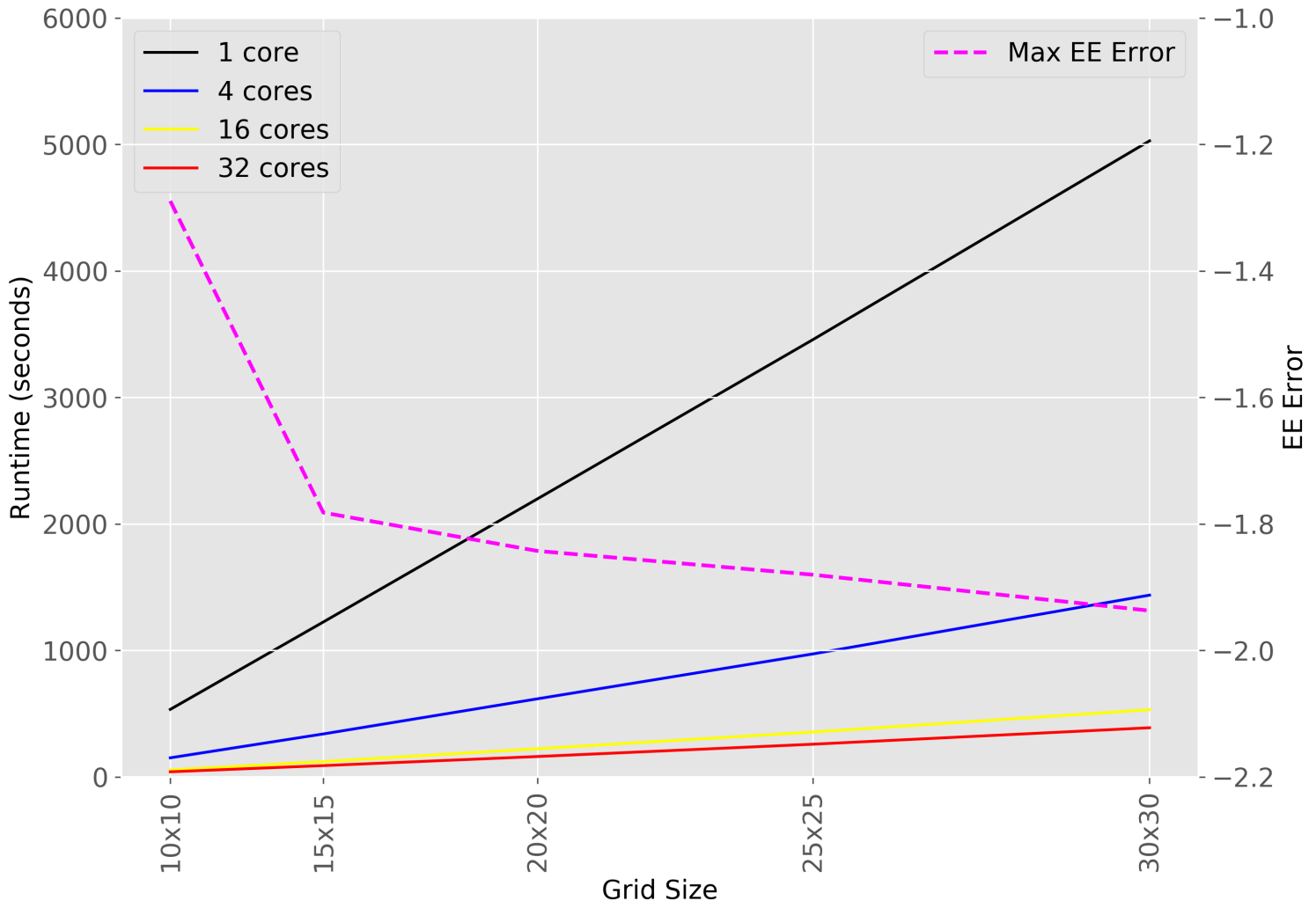
d. Given the obtained policy functions $g^c_t(\hat{a}_t, h_t, e_t) = \hat{c}_t$, $g^s_t(\hat{a}_t, h_t, e_t) = \hat{s}_t$, $g^a_{t+1}(\hat{a}_t, h_t, e_t) = a_{t+1}$, interpolate new policy functions, $g^c_t(a_t, h_t, e_t) = \tilde{c}_t$, $g^s_t(a_t, h_t, e_t) = \tilde{s}_t$, and $g^a_{t+1}(a_t, h_t, e_t) = \tilde{a}_{t+1}$ for each grid point (a_t, h_t, e_t).

e. If $g^a_{t+1}(a_t, h_t, e_t) = \tilde{a}_{t+1} < \underline{a}$, then do Step 3(a) only for the specific (a_t, h_t, e_t) and $a_{t+1} = \underline{a}$. Otherwise, we compute V^a_t and V^h_t from (EC1) and (EC2).

Model 3: EGM Running Time by Cores



Model 3: EGM Running Time and Accuracy by Grids



TECHNICAL ISSUES USING MATLAB & PYTHON IN HPC

Matlab:

- The drawback of the Parallel Computing Toolbox is that it contains digital restrictions that only permit it to operate with 12 cores or fewer on a single computer.
- Matlab Distributed Computing Server costs several thousand dollars depending on the number of nodes licensed.

Python:

- A joint use of numba and multiprocessing packages has a systemic conflict.

Conclusion

- *Efficiently* running a parallel program in HPC environment requires a great amount of engineering effort.
- Parallelization overhead sets an upper bound to the HPC speed-ups.
- A design of HPC-efficient algorithm to reduce the parallelization overhead is essential for speed-ups in HPC.