Making health economic models more efficient in dealing with contemporary complexities

JE. Poirrier[1], J. Maervoet[1], R. Bergemann[2]

[1]Parexel, HEOR Modeling, Wavre, WBR, Belgium;
[2]Parexel International, Basel, Switzerland

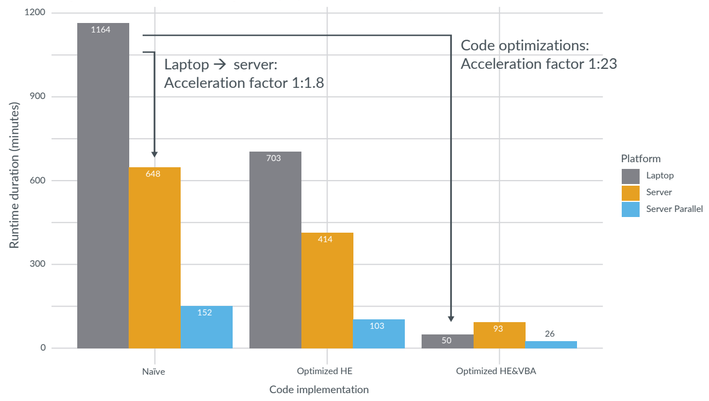
Background

- Health economic models (HEM) are widely used to assess the cost-effectiveness of healthcare interventions and inform decision-making by policymakers and payers[1]. However, their complexity is increasing, as they are being used to analyze more diverse and detailed data sets, more complex interventions and patient pathways, leading to the development of complex analysis frameworks and sensitivity analyses[2]. This poster brings conceptual solutions to these computational issues and illustrates them with examples in MS-Excel.
- First, it is necessary understand and review the model concepts. Often modeling frameworks have requirements and inefficiencies (e.g., keeping a Markov trace) that can lead to computational issues(e.g., out-of-memory execution, inefficient loops). Understanding the modeling framework allows to focus computation on effective parts.
- Second, it is necessary to understand the implemented algorithm, the language, its intricacies, environment and limitations. For instance, Visual Basic for Applications (VBA) code inherits from Excel user-interface features that slow down data processing and can be removed. VBA code is weakly typed, and its type-determination is sometimes problematic, with issues arising "far" from where the initial code is. A common challenge is volatile functions, functions in which the value changes each time the cell is calculated (e.g., OFFSET, INDIRECT, CELL, SUMIF).
- Once the framework is appropriate and the code optimized, a third step is to increase the computing power available for the model. Even without code parallelization, the variety of powerful processors available in commercial cloud offerings makes it easy and cheap for HEMs to run much faster.
- Fourth, code parallelization or parallel distribution of tasks allow to reach linear gain in computation time for tasks that permit (typically: DSA and PSA).

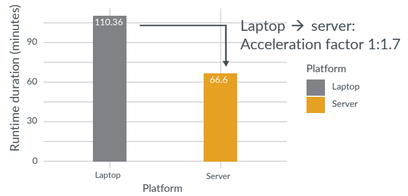
Methods

- All tests were performed on a regular laptop and a server made for computationally-intensive tasks (specifications in Table 1). Both run MS-Windows 10 Enterprise 64 bits, MS-Excel 365 MSO (Version 2208 Build 16.0.15601.20526) 32-bit and R (4.1.1, 64-bit).
- In **Experiment 1**, we run a multi-way DSA from a Markov model for Multiple Sclerosis, 2,880 iterations in a XLSM file.
  - First with a naïve code implementation
  - Then with optimizations in the Markov trace ("OptimizedHE") and in the VBA code (reducing volatile functions, removing any screen interactions, … "OptimizedHE&VBA")
  - Finally, by splitting the task into 6 jobs on 6 Excel instances managed by R (on the server only)
- In **Experiment 2**, we run a PSA from a Markov model for oncology, 10,000 iterations in a XLSM file.
- In **Experiment 3**, we open an XLSB file with 1,000+ volatile OFFSET() function
- All experiments are run on a freshly started system with no other user software running and background processes reduced to a minimum (e.g., OneDrive synchronization stopped).

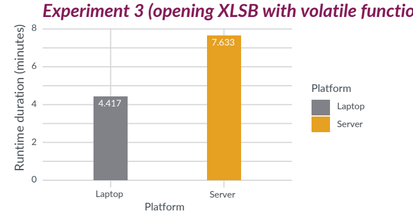
Experiment 1 (Multi-way DSA, 2,880 iterations)



Experiment 2 (PSA, 10,000 iterations)



Experiment 3 (opening XLSB with volatile functions)



Conclusions

- Our benchmarks show a ~2-20 times acceleration depending on the task, when applying model optimization, VBA code optimization and/or switch to a computing server.
- **On a regular laptop, optimizing both the HE model and the VBA code can cut the runtime duration of a multi-way DSA by a factor ~20**
- **Switching from a regular laptop to a computing server can cut the runtime duration of a multi-way DSA or a PSA by a factor ~1.7**
- **Executing parts of a multi-way DSA or a PSA in parallel on cores available on a computing server can nearly linearly accelerate the runtime**
- Volatile functions and optimized Excel VBA code don't take advantage of a non-parallel computing server

Table 1. Test machines specifications

Category	Laptop (2021)	Server (2022)
CPU	Intel Core i5 8365U	Intel Xeon Platinum 8370C
Base frequency	1.6 GHz	2.8 GHz
# cores	4	64
# threads	8	128
RAM	16 Gb	64 Gb
Hard disk	SSD	SSD
GPU	UHD Graphics 620	N.A.
Windows Geekbench.com 5 single-core score	823	1,191
Windows Geekbench.com 5 multi-core score	2,591	38,967

ABBREVIATIONS

DSA: deterministic sensitivity analysis; GPU: graphical processing unit; PSA: probabilistic sensitivity analysis; RAM: random-access memory; SSD: solid-state drive hard disk

REFERENCES

[1] Hollman et al., 2017 (10.1007/s40273-017-0510-8)
[2] Incerti et al., 2019 (10.1016/j.jval.2019.01.003)

Parexel International
2520 Meridian Pkwy
Durham, NC 27713, USA
+1 919 544-3170
www.parexel.com