DFM'2019 PANEL

Execution and Programming Models: Extreme Scale and Beyond

Erik Altman

The 8th IEEE International Workshop on Data Flow Models and Extreme-Scale Computing

QUESTION I

PROGRAM EXECUTION MODEL (PXM) *VS* PROGRAMMING MODEL (PM)

What is the main distinction (as well as relation) between the concepts of PXM *vs* PM ?

- **PM** is the model in which the programmer thinks
- **PXM** is the model run by the machine

- **PM** defines higher-level abstractions than PXM

QUESTION I

PROGRAM EXECUTION MODEL (PXM) *VS* PROGRAMMING MODEL (PM)

What is the main distinction (as well as relation) between the concepts of PXM *vs* PM ?

- Two pairs of definitions are commonly used outside dataflow:

1. **PM** = High-Level Language **PXM** = Assembly Language
2. **PM** = ISA **PXM** = Microarchitecture

- The (1) – (2) distinction is important mainly for machine families (x86, ARM, IBM Z, etc)
  - Dataflow has not had the commercial success for intergenerational compatibility to be an issue ☹
  - *Dataflow:* **PM** = High-Level Language **PXM** = Microarchitecture

QUESTION I

PROGRAM EXECUTION MODEL (PXM) *VS* PROGRAMMING MODEL (PM)

What is the main distinction (as well as relation) between the concepts of PXM *vs* PM ?

Dataflow Example from CAPSL / University of Delaware:

1. **What Are Codelets?**
   - A codelet is a (usually short) sequence of machine instructions that executes until completion.
2. **The Codelet Firing Rule**
   - A codelet can *fire* if all of its dependencies are satisfied.
3. **The Codelet Abstract Machine Model**
   - Describes the mechanisms on which codelets rely to be allocated, stored, and scheduled.
   - Is meant to reflect how future extreme-scale systems will look.
   - We picture a hierarchical machine with compute nodes linked by some kind of interconnect.

- **PM** = (1) + (2)
- **PXM** = (3)

https://www.capsl.udel.edu/codelets.shtml

QUESTION 2

SYSTEM-LEVEL API AND FINE-GRAIN PARALLELISM

- There is a heated discussion and debate on the following vision:
  - "In order to effectively and efficiently exploit the vast parallelism (both at coarse-grain and fine-grain levels) at extreme-scale – we need to break some traditional abstractions at both the PXM and PM levels.
  - This is essential in the design of a systems-level API for future extreme-scale parallel computing systems."
- What is your opinion ?

QUESTION 2

SYSTEM-LEVEL API AND FINE-GRAIN PARALLELISM

To exploit the vast parallelism at extreme-scale – we need to break some traditional abstractions at both the PXM and PM levels.

What is your opinion ?

Ease of programming wins

- Weak *vs* Strong Consistency:
  - Weak may get better performance, but few people can or want to program to it.

Not sure we need to "break" traditional abstractions as much as devise new ones. *Examples:*

- Map-Reduce / Hadoop / Spark enabled massive parallelism and easy programming.
  - At the cost of efficiency in some instances.
- Ideas in my talk on *"Higher-Level Data Types and Algorithms as First-Class Objects"*.

QUESTION 2

SYSTEM-LEVEL API AND FINE-GRAIN PARALLELISM

To exploit the vast parallelism at extreme-scale – we need to break some traditional abstractions at both the PXM and PM levels.

What is your opinion ?

Extreme-scale has always had strong focus on efficiency:

- Machines are expensive one-offs
  - → They require maximum utilization to justify their acquisition cost.
- Those economics drive different tradeoffs in ease-of-use than the commercial market.

Is it time for a more COTS-based approach with a goal of getting extreme-scale to adopt it?

- 1990s Examples: **RAID**

  NOW – Network of Workstations

QUESTION 3

ON THE PROGRAMMABILITY OF DATAFLOW MODELS

- There have been significant concerns that
  - “The dataflow/codelet community has always claimed their model is more productive;
  - However more recent work with task parallelism and the recent OCR project tried working with these types of models, and the scientific application community actually found them less productive.”
- What is your observation/opinion ?

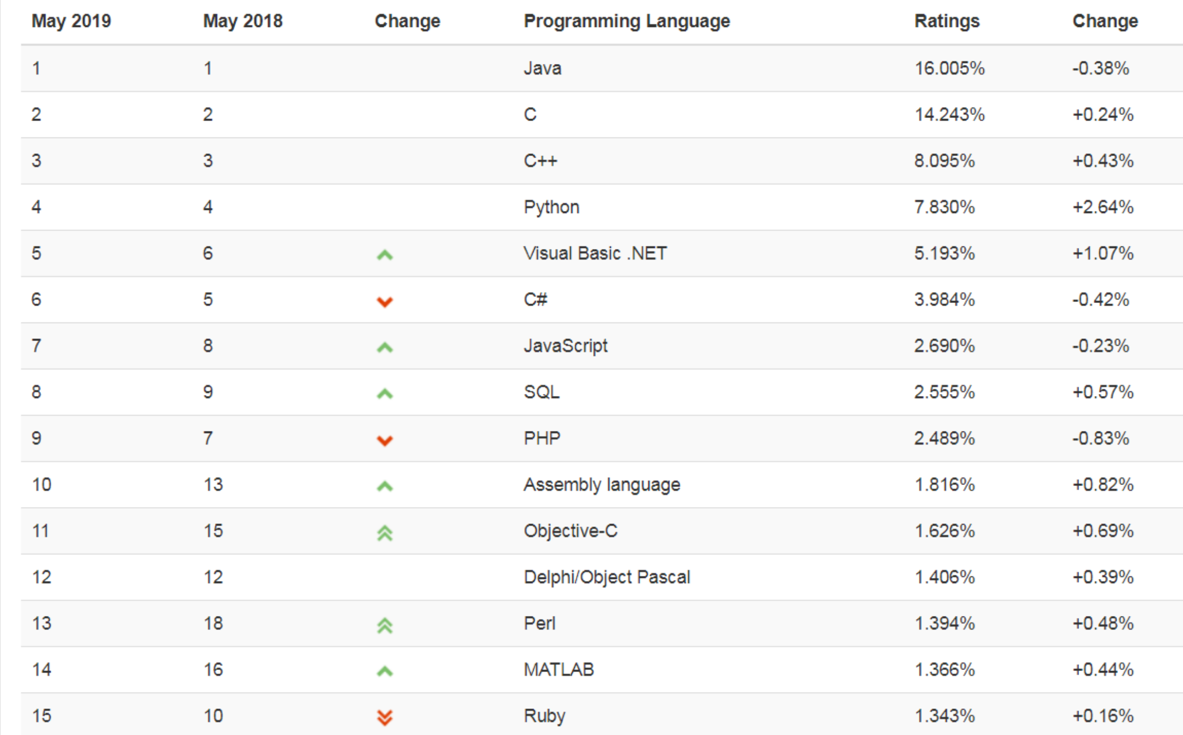
QUESTION 3

ON THE PROGRAMMABILITY OF DATAFLOW MODELS

The scientific application community actually found [OCR / dataflow] less productive.

What is your observation/opinion ?

- People are more productive in familiar environments.
- Dataflow / OCR is less familiar to most than traditional approaches like MPI and OpenMP.



May 2019	May 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.005%	-0.38%
2	2		C	14.243%	+0.24%
3	3		C++	8.095%	+0.43%
4	4		Python	7.830%	+2.64%
5	6	▲	Visual Basic .NET	5.193%	+1.07%
6	5	▼	C#	3.984%	-0.42%
7	8	▲	JavaScript	2.690%	-0.23%
8	9	▲	SQL	2.555%	+0.57%
9	7	▼	PHP	2.489%	-0.83%
10	13	▲	Assembly language	1.816%	+0.82%
11	15	▲▲	Objective-C	1.626%	+0.69%
12	12		Delphi/Object Pascal	1.406%	+0.39%
13	18	▲▲	Perl	1.394%	+0.48%
14	16	▲	MATLAB	1.366%	+0.44%
15	10	▼▼	Ruby	1.343%	+0.16%

Virtually all the most popular languages are old:

Java, C, C++, Python, Visual Basic, C#, JavaScript, SQL, PHP, Assembly, Objective-C, Pascal, Perl, MATLAB, Ruby

https://www.tiobe.com/tiobe-index

QUESTION 3

ON THE PROGRAMMABILITY OF DATAFLOW MODELS

The scientific application community actually found [OCR / dataflow] less productive.

What is your observation/opinion ?

- It would also be interesting to see productivity comparisons of students.
  - *Students previously unfamiliar with historical techniques or dataflow / OCR*
- But other programming paradigms have gained popularity → Ties back to Question 2 Response:
  - *Not sure we need to "break" traditional abstractions as much as devise new ones:*
    - a la Map-Reduce / Hadoop / Spark, RAID, and NOW
  - *Have we been too constrained and low-level in how we expose dataflow for extreme scale?*