Extra-P: Application Performance Modelling and Application Mapping



TECHNISCHE UNIVERSITÄT DARMSTADT

Alexander Geiß, Technical University of Darmstadt

Eile View Plots Model Help									
Selection		Ø	3			Modeler	0 1		
Model:	efault Model	•	Line graph 🗙	Measurement points X		Model name:	New Model		
- House -									
Metric: ti	Metric: time					Model mean	Model median		
Sev: Callp	ath	Com Value RSS Adj. R ²		Measurem	nent Points	Model generato	r:		
	main	3.954x10 ⁻⁰⁵ + 3.441x10 ⁻⁰⁶ * log ₂ (n) - 8.925x10 ⁻⁰⁶ * log ₂ (gpu) * log ₂ (n) 3.160x10 ⁻¹⁰ 0.421			cycleTracking(MonteCarlo*)	Default	•		
	initMC()	1.092x10 ⁻⁰⁴ 2.675x10 ⁻⁰⁷ 1							
	 cycleTracking() 	0.127 - 8.448x10 ⁻⁹⁷ * n * log₂(n) + 8.368x10 ⁻⁹⁷ * gpu ¹ ∕ ⁴ * n * log₂(n) 0.620 0.994				 Advanced 	options		
	cudaDeviceSy	0.284 + 0.214 * log ₂ (gpu) + 5.134x10 ⁻¹² * n ^{3/2} * log ₂ (n) ^{2.0} 0.132 0.960							
	ParticleVault:	-0.036 - 1.460x10 ^{-ve} * n ^{1/4} * log ₂ (n) ²⁻⁰ + 6.548x10 ^{-ve} * gpu ^{1/3} * n ^{1/4} 1.318x10 ^{-ve} 0.941				G	enerate models		
	ParticleVaultC	1.147x10 ⁻⁰⁵ - 4.770x10 ⁻¹⁶ * n * log ₂ (n) ²⁻¹⁶ + 4.736x10 ⁻¹⁶ * gpu ^{1/2} * n 1.551x10 ⁻¹⁴ 0.986							
	MC_Particle	0.035 - 1.274x10 ** n *** log2(n) + 6.225x10 ** gpu * n *** log2 0.010 0.983							
	MC_Particle	0.331X10 - 1.37/X10 - n + 1.231X10 - gpu - n - 3.028X10 - 0.993							
	MC Particle	$-1.221\times10^{-93} + 4.327\times10^{-97} \times n + 2.022\times10^{-97} \times n + 2.022\times10^{-97} \times n + 1.146\times10^{-93} - 0.007$							
	MC Particle	5 332 v 10 ⁻⁰⁶ + 7 950 v 10 ⁻⁰⁶ + log ₂ (mu) 2 030 v 10 ⁻⁰⁶ + 0.000							
	MC Particle	5 537x 10 ⁻⁰⁴ + 7.960x 10 ⁻⁰⁷ * log ₂ (gpu) * n ^{1/2} * log ₂ (n) 1.541x 10 ⁻⁰⁴ 0.957							
	MC Particle	-5.090x10 ⁻⁰⁵ + 5.317x10 ⁻⁰⁵ * apu ^{1/4} 9.756x10 ⁻¹¹ 0.000			U T				
	NVTX Range:	1.087x10 ⁻⁰³ - 4.702x10 ⁻¹⁰ * n * log ₂ (n) ²⁻⁰ + 4.671x10 ⁻¹⁰ * gpu ^{1/4} * n 1.518x10 ⁻⁰⁴ 0.986			1 6 <u>é</u>				
	std::chrono::d	3.284x10 ⁻⁰³ - 1.629x10 ⁻⁰⁹ * n * log ₂ (n) ²⁻⁰ + 1.622x10 ⁻⁰⁹ * gpu ^{1/4} * n 1.828x10 ⁻⁰³ 0.986							
	std::common	1.823x10 ⁻⁰³ - 5.312x10 ⁻⁰⁸ * n * log ₂ (n) + 5.343x10 ⁻⁰⁸ * gpu ^{1/4} * n * l 5.075x10 ⁻⁰³ 0.987							
	std::enable_if	4.438x10 ⁻⁶⁴ - 4.139x10 ⁻⁶⁸ * n * log ₂ (n) + 4.176x10 ⁻⁶⁸ * gpu ¹ / ⁴ * n * l 3.032x10 ⁻⁶³ 0.988							
	ParticleVaultC	2.073x10 ⁻⁰⁶ 1.606x10 ⁻¹¹ 1							
	ParticleVault:	4.534x10 ⁻⁰³ - 8.131x10 ⁻¹¹ * n ⁴ / ³ * log ₂ (n) + 7.520x10 ⁻¹¹ * gpu ¹ / ³ * n ⁴ 1.701x10 ⁻⁰⁴ 0.983	3						
	ParticleVaultC	9.986x10 ⁻⁰⁴ - 4.421x10 ⁻¹⁰ * n * log ₂ (n) ²⁻⁰ + 4.388x10 ⁻¹⁰ * gpu ^{1/4} * n 1.258x10 ⁻⁰⁴ 0.987							
	ParticleVaultC	9.468x10 ⁻⁶⁴ - 4.393x10 ⁻¹⁶ * n * log ₂ (n) ²⁻⁶ + 4.364x10 ⁻¹⁶ * gpu ^{1/4} * n 1.298x10 ⁻⁶⁴ 0.986							
	ParticleVaultC	5.628x10 ^{-0s} + 3.848x10 ^{-0s} * log ₂ (gpu) * n * log ₂ (n) 8.061x10 ⁻⁰⁴ 0.978			600000				
	ParticleVaultC	7.377x10 - 3.874x10 n * log ₂ (n) + 3.859x10 * gpu * n 1.032x10 0.986			1 500000				
	ParticleVaultC	$1.0/0x10^{-9} - 9.525x10^{-9} n^{-1} \log_2(n) + 9.481x10^{-9} gpu^{-7} n^{-1} l 1.53/x10^{-0} 0.988$			400000				
	ParticleVaultC	0.023 - 1.200-10 ⁻⁰⁹ * = * l== (=) ²⁻⁰ - 4.070-10 ⁻¹¹ * ==, ^{2/3} * l== (== -2.400-10 ⁻⁰⁸ 0.001							
	SondOussions	$2.475 \times 10^{-03} = 9.763 \times 10^{-10} \times n \times \log_2(n)^{2-0} + 9.573 \times 10^{-10} \times n \times 10^{1/4} \times n = 1.196 \times 10^{-04} - 0.007$		2.5	300000				
	SendQueuerro	$1.152 \times 10^{-03} = 4.347 \times 10^{-10} \times n \times \log_2(n)^{2-0} + 4.309 \times 10^{-10} \times n \times 1.290 \times 10^{-04} \cdot 0.986$		5.0 7.5	200000 n				
	ThreadBlockL	2,965x10 ⁻⁰⁵ - 1,338x10 ⁻⁰⁴ * log ₂ (gpu) + 6,481x10 ⁻⁰⁷ * log ₂ (gpu) * log ₄ 2,112x10 ⁻⁰⁹ 0,993		10.0	100000				
	CycleTracking	5.218x10 ⁻⁰⁴ - 2.738x10 ⁻⁰⁴ * log ₂ (gpu) + 2.472x10 ⁻⁰⁶ * log ₂ (gpu) * n ^{1/2} 2.484x10 ⁻⁰⁷ 0.996		12.5 15.0					
	getExecution	4.123x10 ⁻⁹⁶ - 1.202x10 ⁻⁹⁷ * log ₂ (n) 3.041x10 ⁻¹² 0.000		gpa	17.5 0				
4	NB/TV Deserve	1 47310 ⁻⁰⁴ 7 40110 ⁻¹¹ * * I /							
	•	Show model Show parameters							
			E						
gpu	1 🗘 🗋								
n	1		Graph Limite			20			
			oraph cinits						
Color Info		0) X-axis	jpu	• max. 19,20				
0.000		0.649	Y-axis r	1	• max. 600000,00				

















We need to find scaling issues before they occur We need: a model for performance behavior



Performance model



Formula that expresses a relevant performance metric as a function of one or more execution parameters



Automatic empirical performance modelling with Extra-P







Human-readable, multi-parameter performance models

$$f(x_1, \dots, x_m) = \sum_{k=1}^n c_k \prod_{l=1}^m x_l^{i_{kl}} \cdot \log_2^{j_{kl}}(x_l)$$

Modelling process



TECHNISCHE UNIVERSITÄT DARMSTADT

Performance model normal form



Assumptions & limitations



- Scaling behaviour expressible with performance model normal form
- Only one scaling behaviour for all the measurements; no jumps
- Some MPI collective operations switch their algorithm
 - results in bad models
- Example: red model tries to model measurements of different algorithms
 - First 4 points one function
 - Last 4 points another function (linear)



Modelling application requirements





Lulesh

Models represent per-process effects

- p number of processes
- n problem size per process

Requirement	Metric	Model
Computation	#FLOPs	$10^5 \cdot n \cdot \log(n) \cdot p^{0.25} \cdot \log(p)$
Communication	#Bytes sent & received	$10^3 \cdot n \cdot p^{0.25} \cdot \log(p)$
Memory access	#Loads & stores	$10^5 \cdot n \cdot \log(n) \cdot \log(p)$
Memory footprint	#Bytes used	$10^5 \cdot n \cdot \log(n)$
Memory locality	Stack distance	Constant





UPCOMING FEATURES

GPU Applications







Generating GPU models





Usage

Has the GPU version similar scaling behaviour? Has the GPU version similar or better performance?



- Is runtime of CPU only > host computation + data transfer time + runtime on accelerator?
 - Comparison of CPU app model with models for host computation, data transfer and kernels
- How much work can the CPU do, while the GPU is doing the offloaded work?
 - Synchronization model
- Is the GPU well utilized?
 - Idle time model







How to map an app onto MSA systems?





How to map an app onto MSA systems?





How to map an app onto MSA systems?





How to find a good mapping? Computation Computation Communication

9/3/2021 | Department of Computer Science | Laboratory for Parallel Programming | Alexander Geiß | 20

How to map an app onto MSA systems?



Objectives



Design a strategy for mapping application parts to the MSA modules

Create support for model comparison Create portable performance models Use these models to determine best target modules

Reduce user involvement, if possible

Summary



- Applications can exhibit unwanted performance behaviour when scaling up
- Performance models help to find issues before they occur
 - Laborious to do by hand
 - Extra-P automates this step
- Extra-P will also support GPUs
 - Helps checking for optimization opportunities and unwanted behaviour
- Extra-P will assist in mapping of applications onto MSA systems

Check your app with Extra-P before scaling up!

Find it on GitHub: https://github.com/extra-p/extrap