JIT Compilation Policy For Modern Machines

Prasad A. Kulkarni

OOPSLA'11

Presentation: Piotr Górski

JIT compilation

a method to improve the runtime performance of computer programs written in languages supporting „compile-once, run-anywhere" model for code generation and distribution

Problems

- JIT compilation contributes to the overall execution time of the application
- Compilation policies need to carefully tune **if** and **when** different program regions are compiled to achieve the best performance

Existing policies

- Selective compilation
- Online profiling to detect ***hot methods***
  - Method counters
  - Interrupt-timer-based sampling
  - Assumption that current hot methods will remain hot in the future
  - Delays
    - Time to reach the compilation threshold
    - Time spent in compilation queue
- Proposed approaches
  - Offline profiling
  - Dynamically determining loop iteration bounds to predict future hot methods

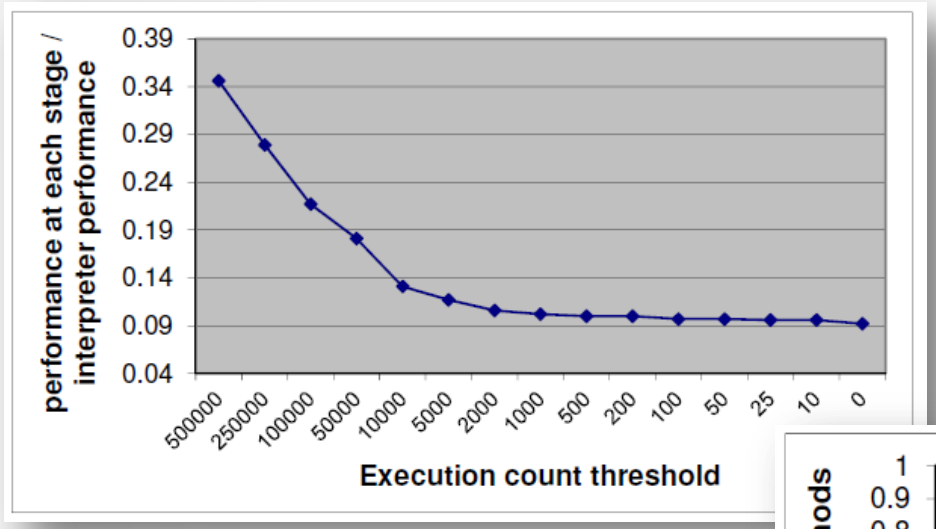
Contribution of the paper

- Quantifying the impact of altering **if** and **when** methods are compiled on application performance
- Demonstrating the effect of multiple compiler threads on average program performance for single-core machines
- Explaining the impact of different JIT compilation strategies on multi-core machines
- Showing the benefit of prioritizing method compiles on program performance

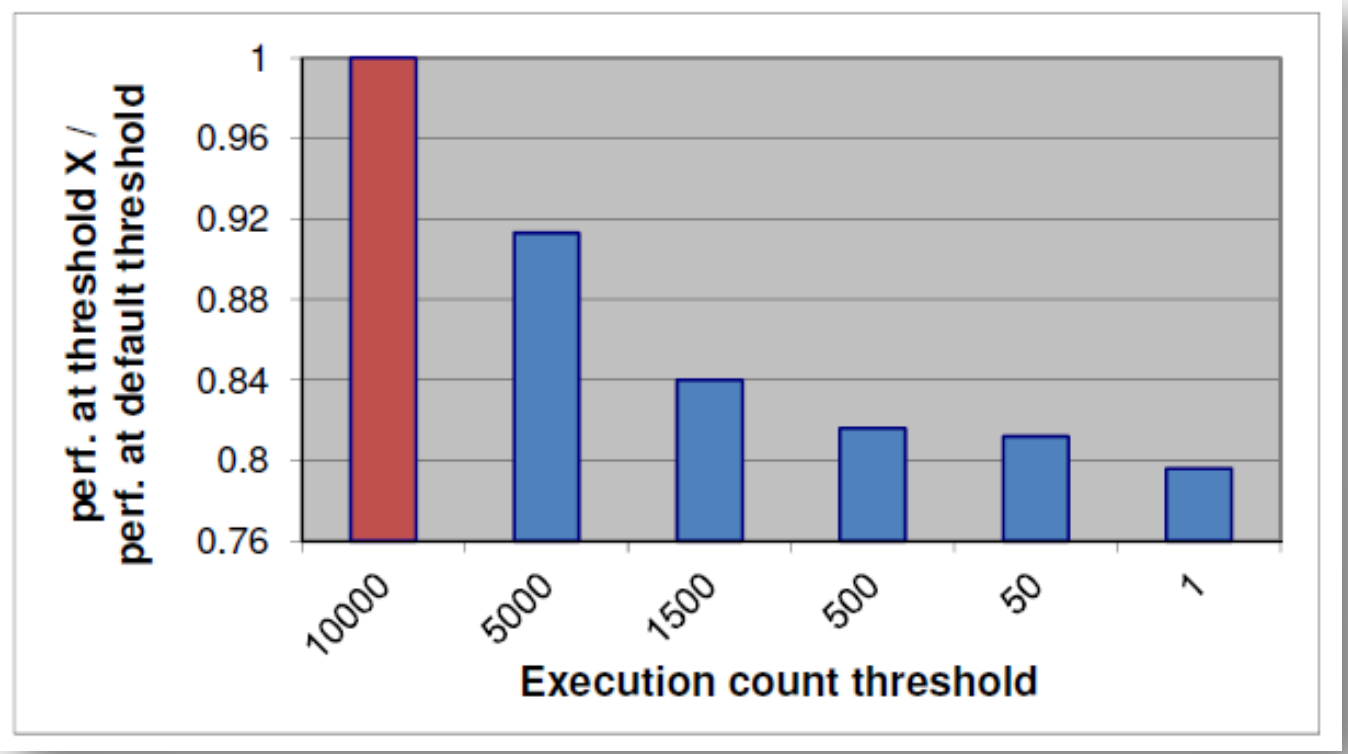
Benchmarks

SPECjvm98		SPECjvm2008		DaCapo-9.12-bach	
Name	#Methods	Name	#Methods	Name	#Methods
_201_compress_100	517	compiler.compiler	3195	avrora_default	1849
_201_compress_10	514	compiler.sunflow	3082	avrora_small	1844
_202_jess_100	778	compress	960	batik_default	4366
_202_jess_10	759	crypto.aes	1186	batik_small	3747
_205_raytrace_100	657	crypto.rsa	960	eclipse_default	11145
_205_raytrace_10	639	crypto.signverify	1042	eclipse_small	5461
_209_db_100	512	mpegaudio	959	fop_default	4245
_209_db_10	515	scimark.fft.small	859	fop_small	4601
_213_javac_100	1239	scimark.lu.small	735	h2_default	2154
_213_javac_10	1211	scimark.monte_carlo	707	h2_small	2142
_222_mpegaudio_100	659	scimark.sor.small	715	jython_default	3547
_222_mpegaudio_10	674	scimark.sparse.small	717	jython_small	2070
_227_mtrt_100	658	serial	1121	luindex_default	1689
_227_mtrt_10	666	sunflow	2015	luindex_small	1425
_228_jack_100	736	xml.transform	2592	lusearch_default	1192
_228_jack_10	734	xml.validation	1794	lusearch_small	1303
				pmd_default	3881
				pmd_small	3058
				sunflow_default	1874
				sunflow_small	1826
				tomcat_default	9286
				tomcat_small	9189
				xalan_default	2296
				xalan_small	2277

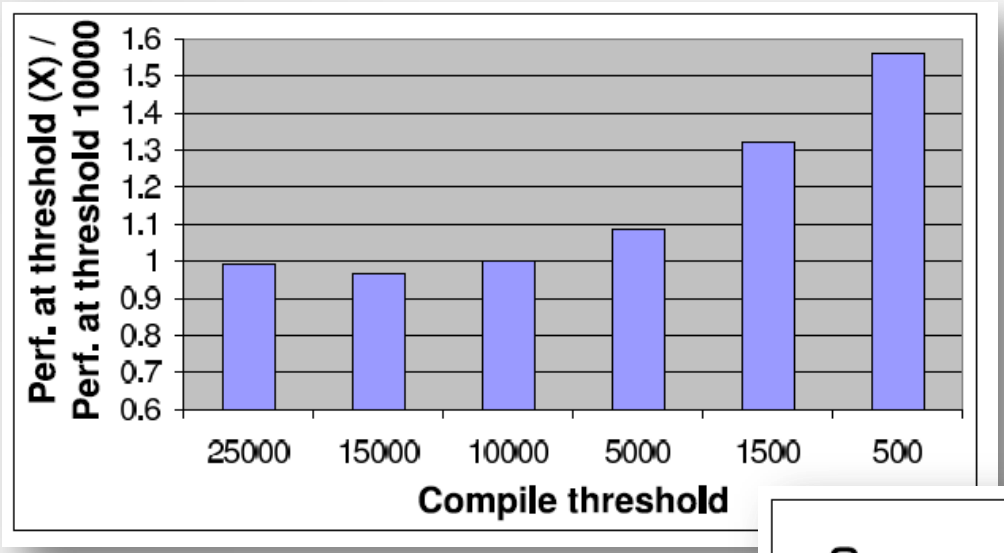
Effect of JIT compilation

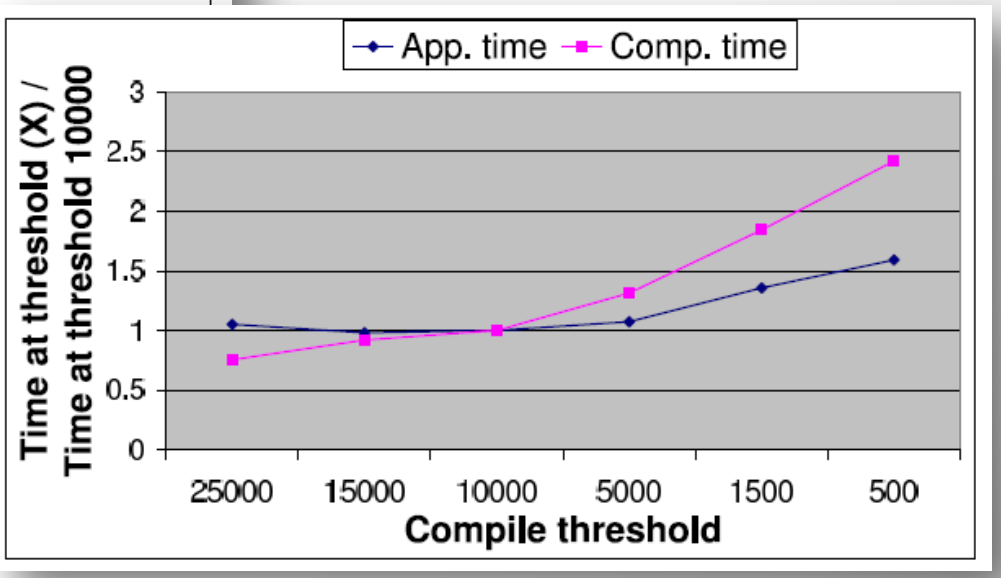


Effect of early compilation



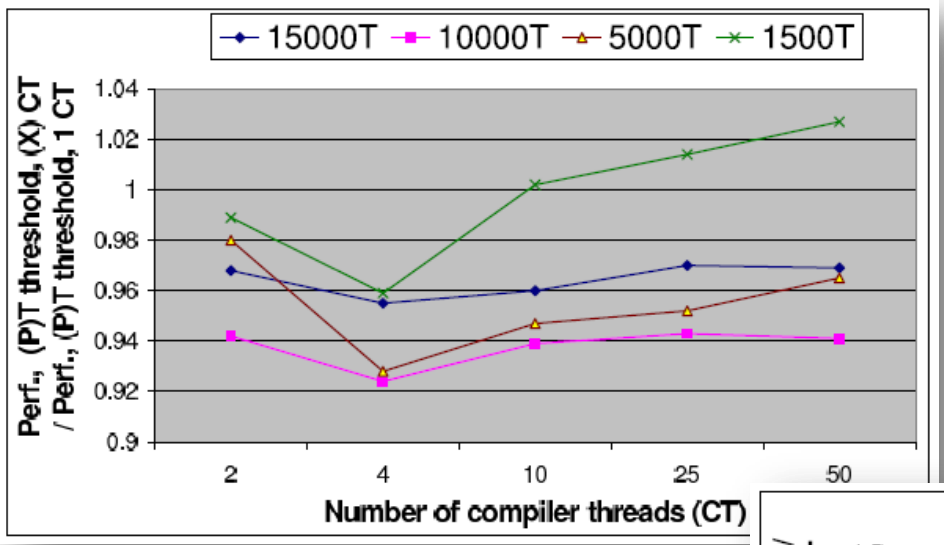
JIT compilation on single-core machines

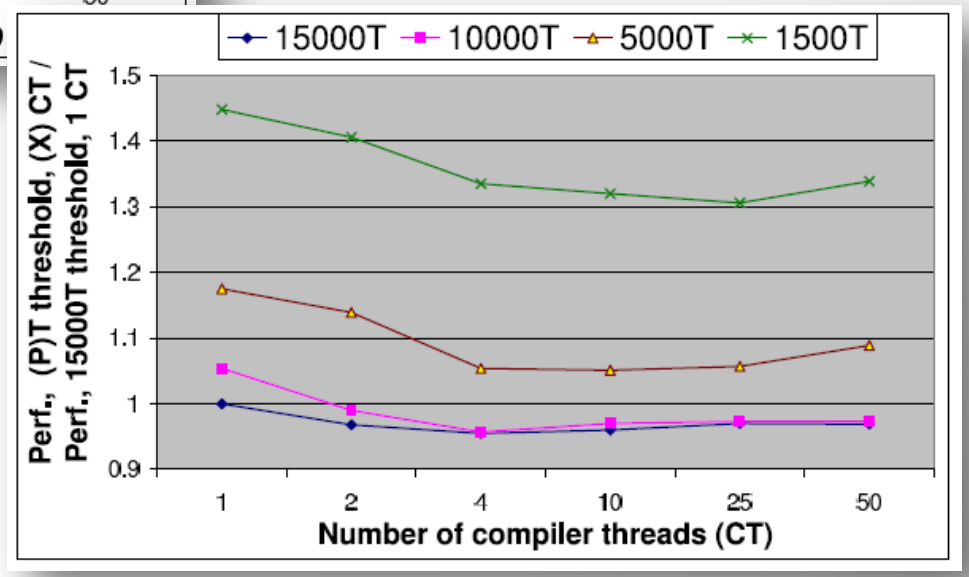




One thread

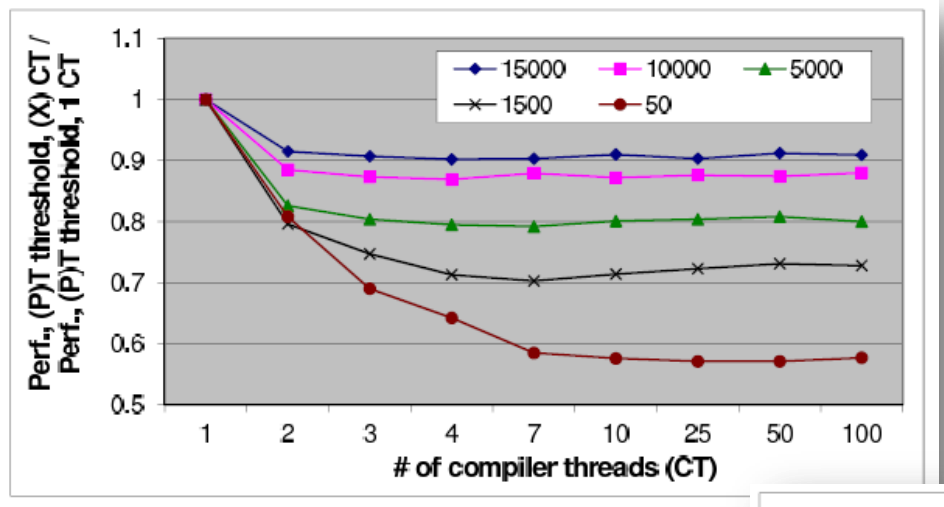
JIT compilation on single-core machines

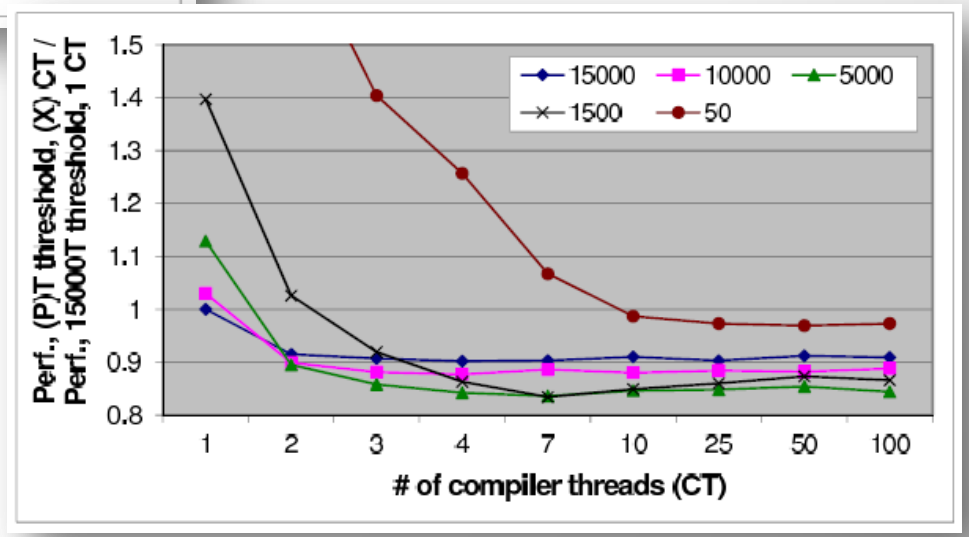




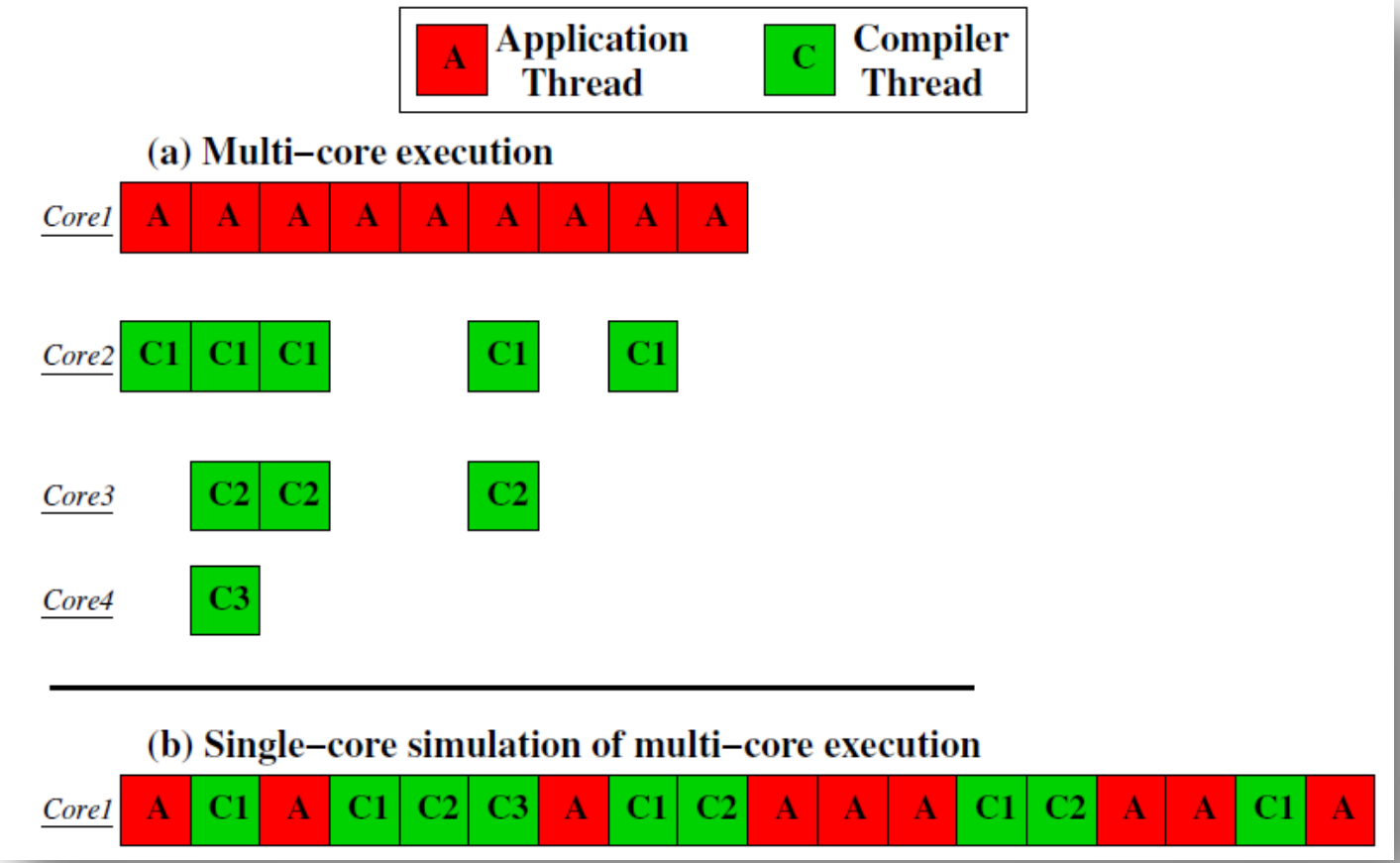
Multiple threads

JIT compilation on multi-core machines

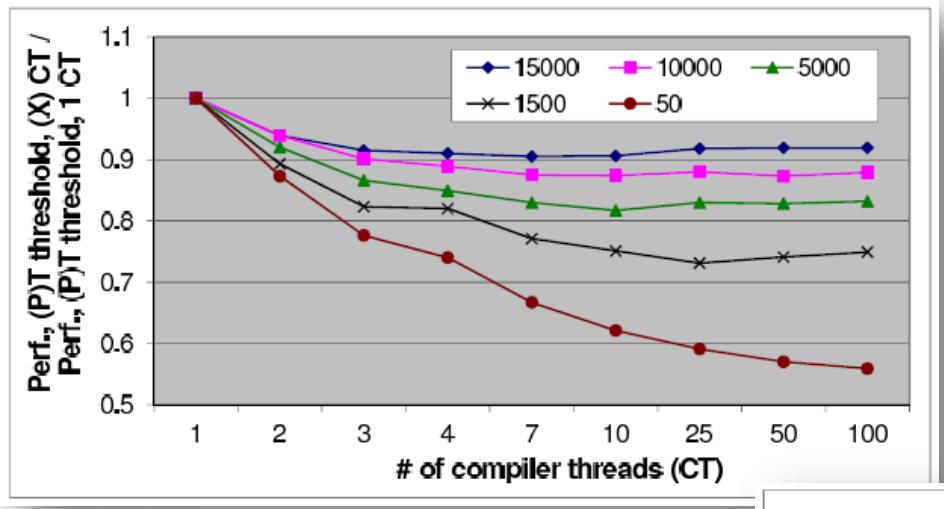


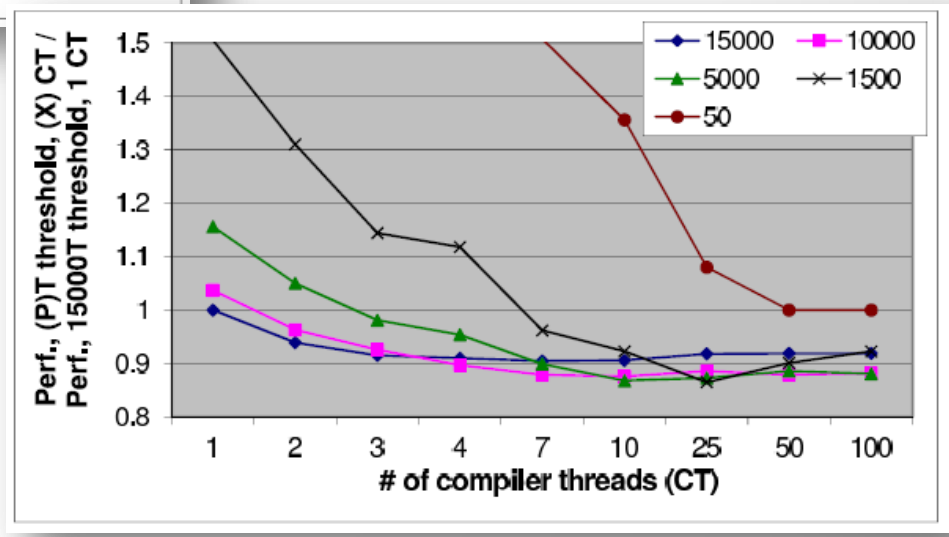


Simulation of multi-core VM execution on single-core processor

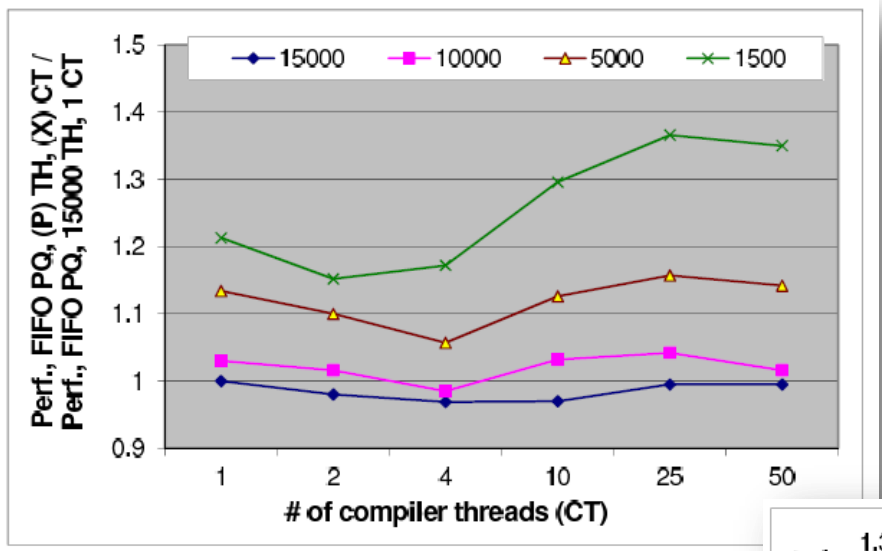


JIT compilation on many-core machines

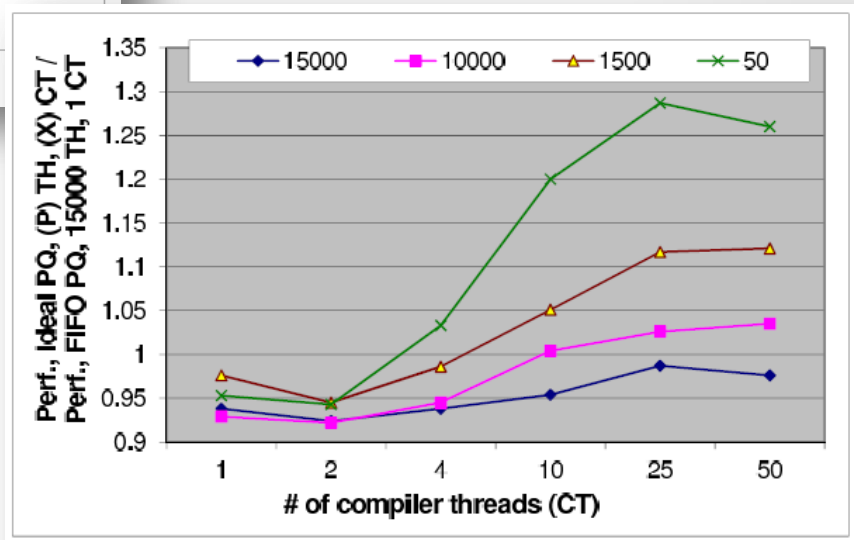




Priority-based compiler queue



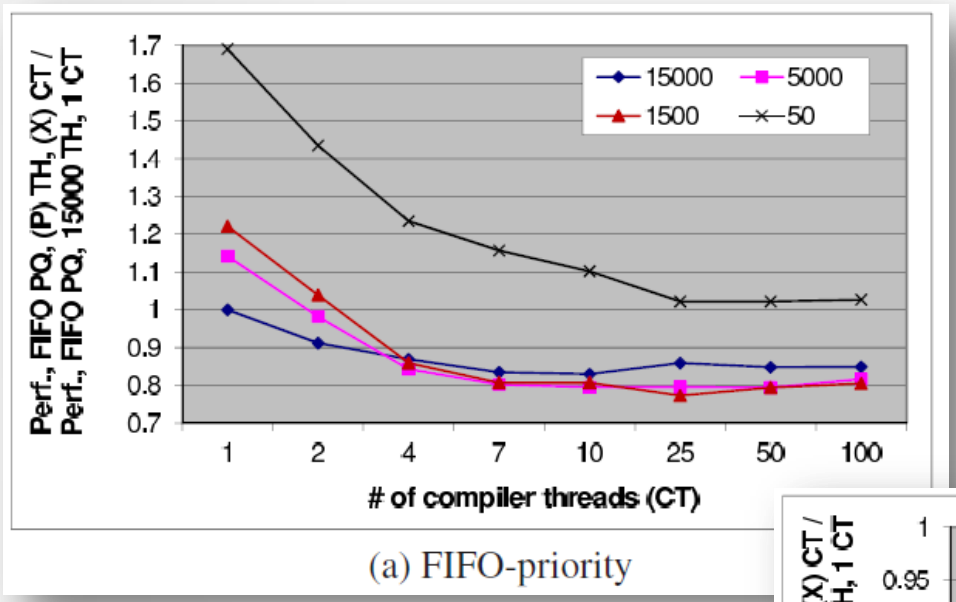
(a) FIFO-priority



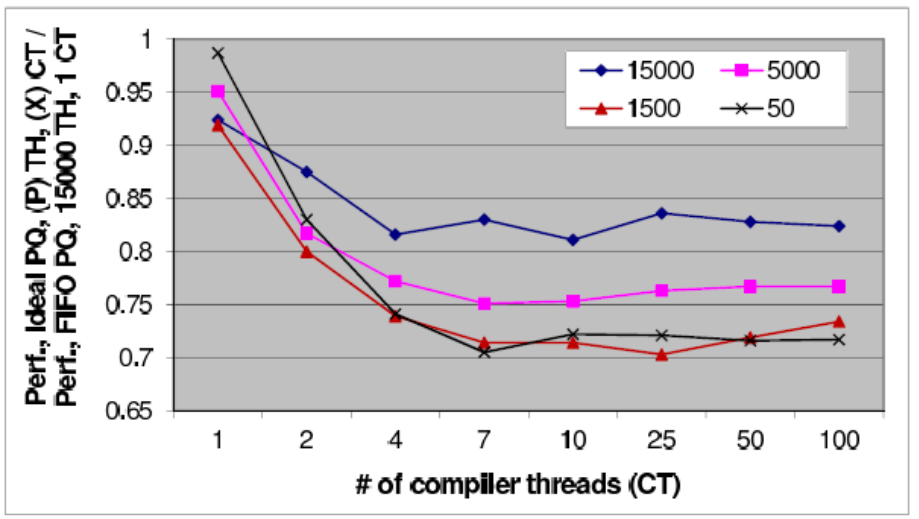
(b) Ideal Priority

single-core machine configuration

Priority-based compiler queue



(a) FIFO-priority



(b) Ideal Priority

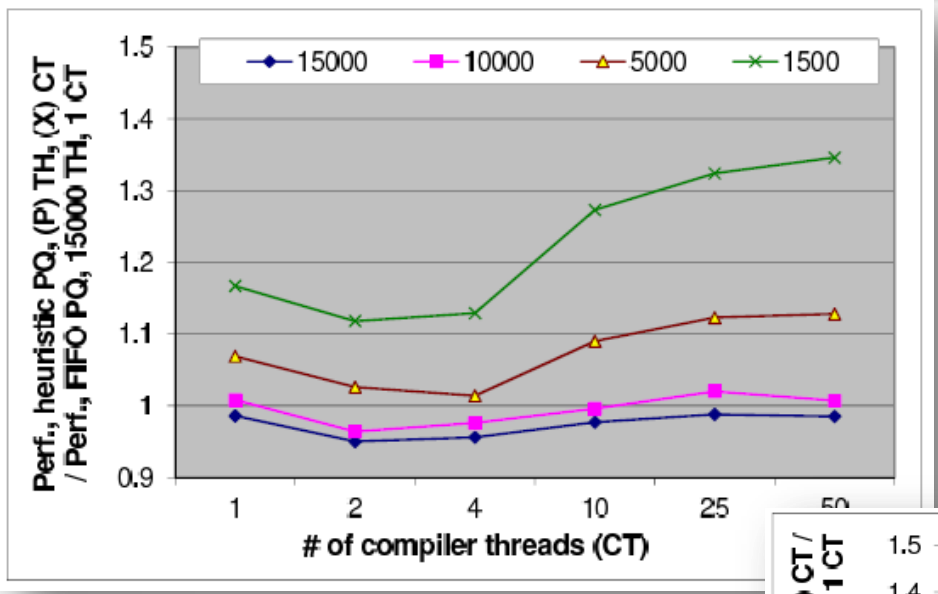
many-core machine configuration

Priority-based compiler queue heuristic approach

- Offline profiling is rarely acceptable
- Need to devise a dynamic priority scheme

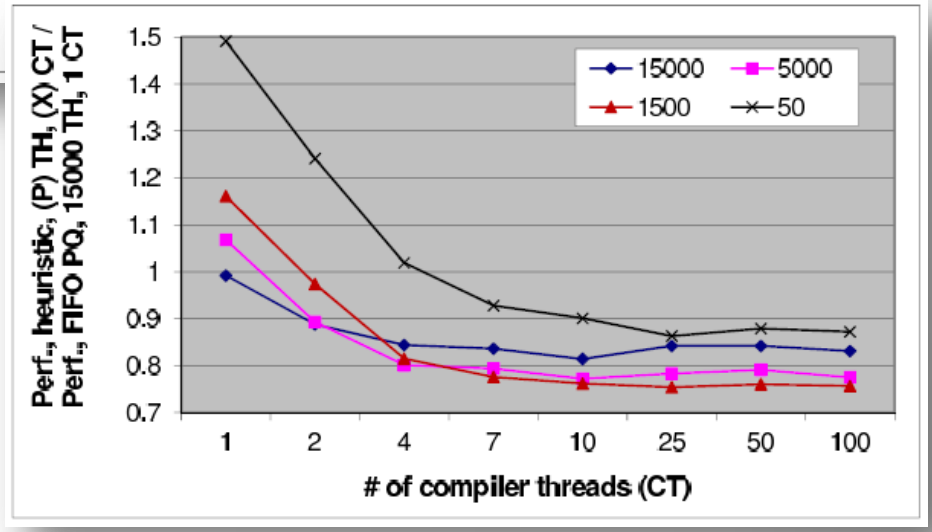
$$priority = \frac{method\ execution\ count}{current\ global\ count - X}$$

Priority-based compiler queue heuristic approach



← single-core machine configuration

many-core machine configuration →



Conclusions

- On single-core machines the same compilation threshold achieves the best overall program performance with a single and multiple compiler threads, and regardless of the priority queue algorithm.
- Multi-core and many-core hardware can enable more aggressive JIT compilation policies, which can be benefactory to performance …
- … but achieving such benefits requires accurate assignment of method priorities.

Thank you