Task Inlining in Phylanx

Bibek Wagle

November 7, 2019

Task Inlining[1]

- Parent task completes the work assigned for the child task
- Child task is never created, hence save the cost of creation and management
- Increases granularity of the parent tasks
- Aggressive inlining results in lost opportunity for parallelism
- Infrequent inlining results in unnecessary overheads
- Delicate balance is needed

[1] B. Wagle, M. A. H. Monil, K. A. Huck, A. D. Malony, A. Serio, H. Kaiser: Runtime Adaptive Task Inlining on Asynchronous Multitasking Runtime Systems. ICPP 2019: 76:1-76:10

Task Inlining Threshold

- How much work should a task contain in order to warrant creating a new task for it ?
- Inlining Threshold
  - Tasks with work smaller than the threshold does not warrant creating a new task
  - Task with work larger than the threshold has enough work to amortize the cost of overhead

Performance Impact of Task Inlining

- Applications
  - Logistic Regression
  - Alternating Least Squares
- Machines
  - Rostam Cluster at LSU
  - Intel Skylake, Sandybridge, Haswell
  - AMD Bulldozer

Performance Impact of Task Inlining

Table 1: Specifications of machines used

Make	Intel	Intel	Intel	AMD
Arch.	Sandybridge	Haswell	Skylake	Bulldozer
CPU	E5-2450	E5-2660v3	Gold 6148	6272
Cores	8	10	20	16
Frequency	2.1GHz	2.60GHz	2.4GHz	2.1GHZ

Logistic Regression

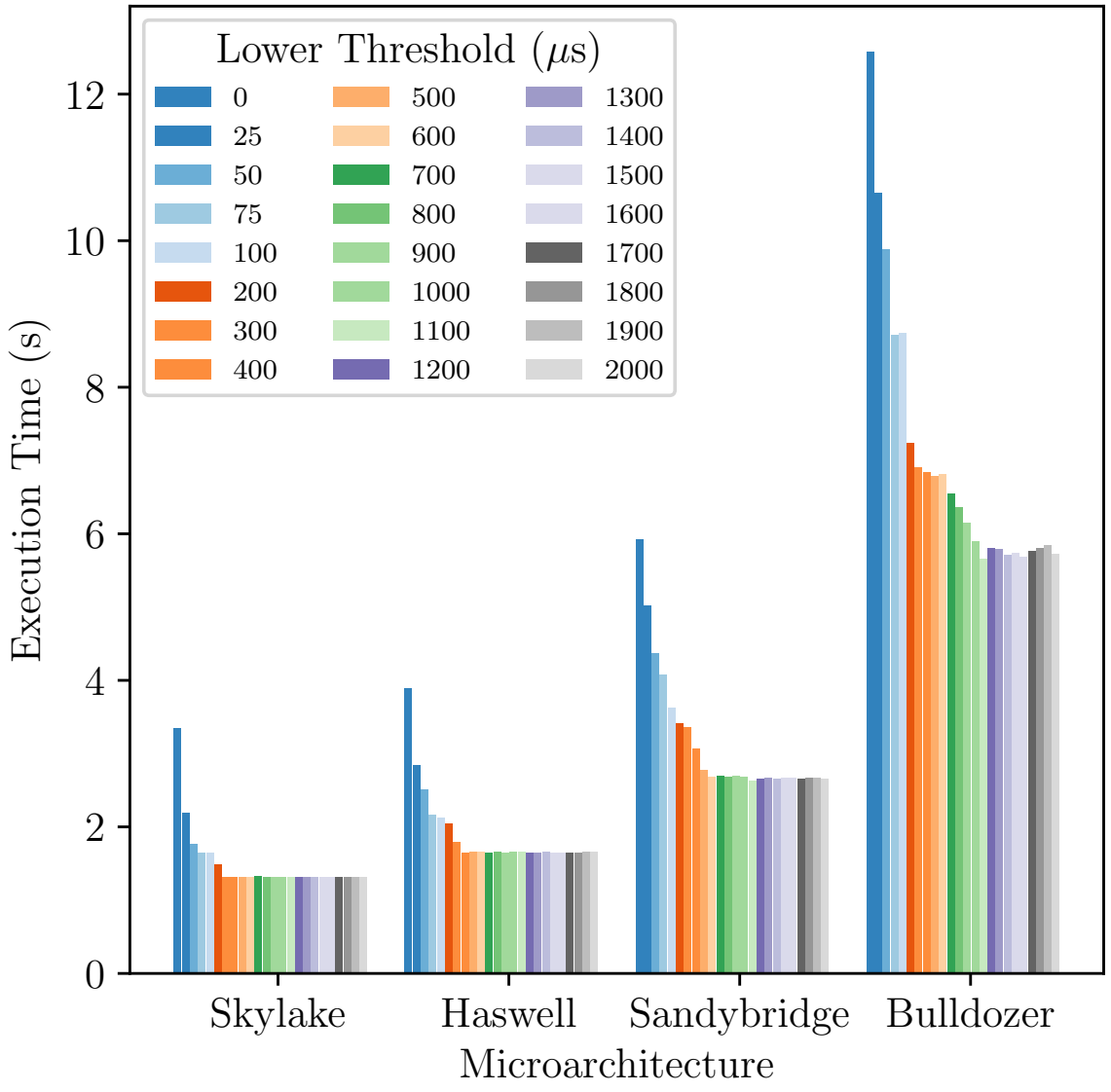

Figure 1: Execution time for LRA on eight threads

Alternating Least Squares

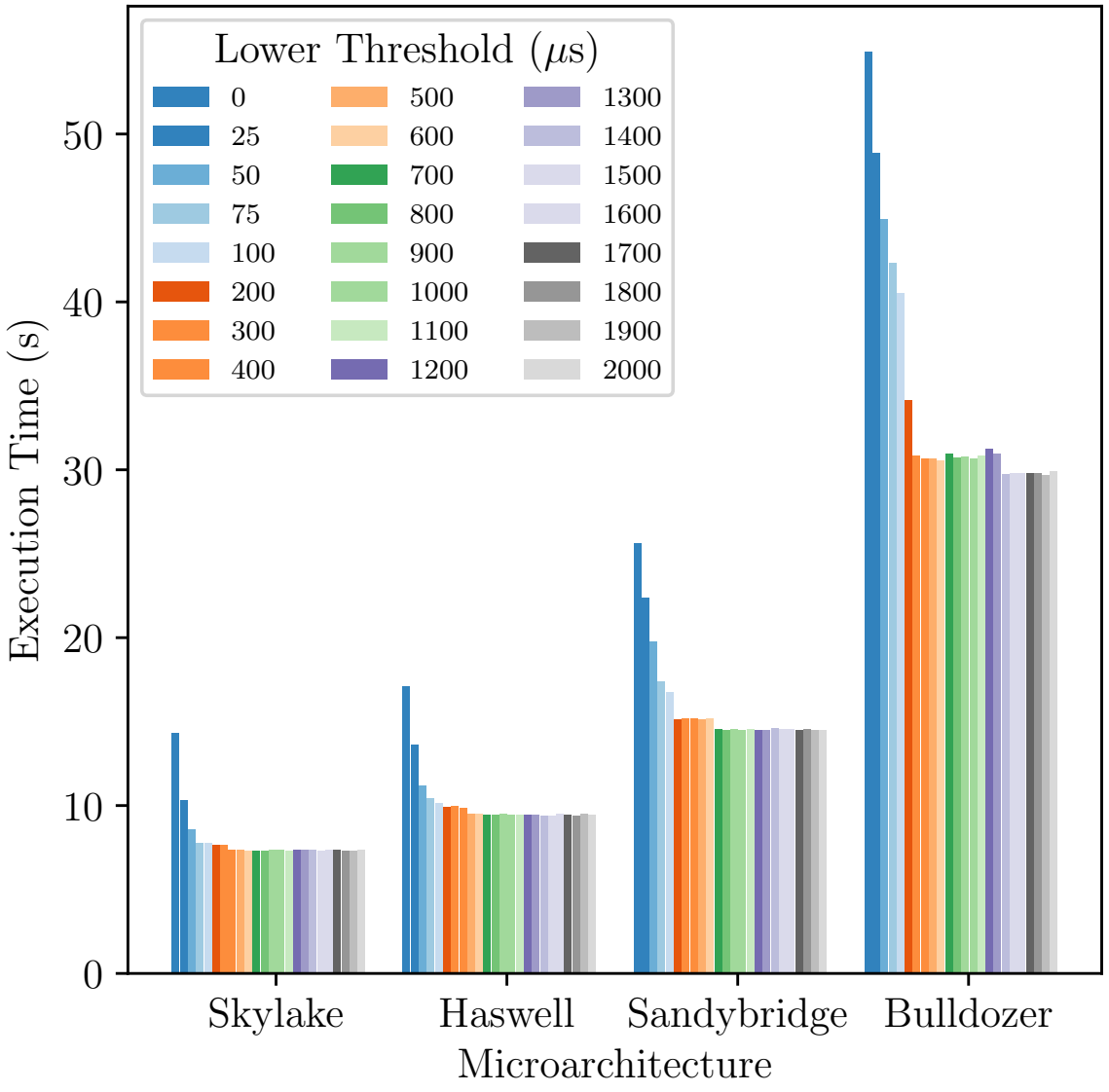

Figure 2: Execution time for ALS on eight threads

APEX Policy

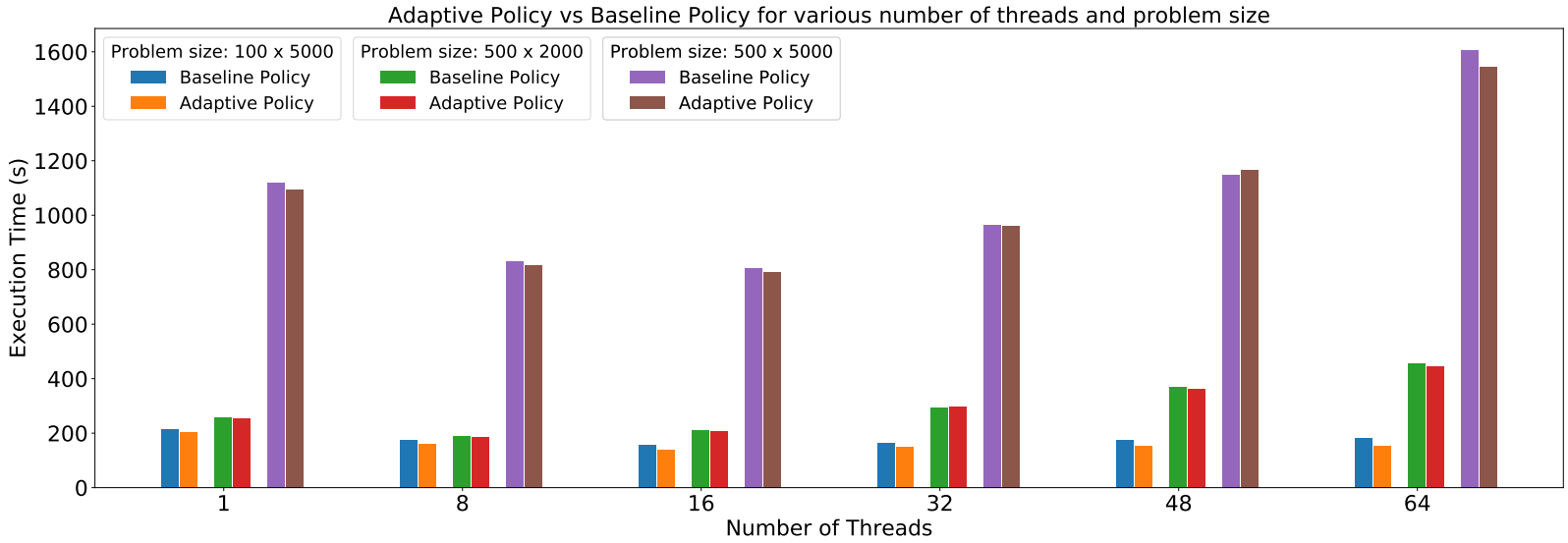

Figure 3: Comparison of the Adaptive APEX and Baseline (Threshold set at 350μs) policies on Bulldozer node for the Alternating Least Squares example for an input size of 100 x 5000, 500 x 2000 and 500 x 5000 elements and various number of threads.

Overheads of HPX-Future

Let t_{oh} be the overhead per HPX task and t_{thres} minimum acceptable threshold beyond with improvement starts tapering off, then,

$$\lambda_{min} = \frac{t_{thres}}{t_{oh}} \tag{1}$$

where λ_{min} is the minimum amount of work necessary per task in order to amortize the cost of overheads.

Summary

- Task inlining improved the execution time of our test application compared to fully asynchronous execution
- Improvement from task inlining tapers off after certain value for inlining threshold
- The threshold at which improvement starts tapering off is different for different processors
- We used APEX for selecting the correct inlining threshold for a particular architecture

PhySL vs Python

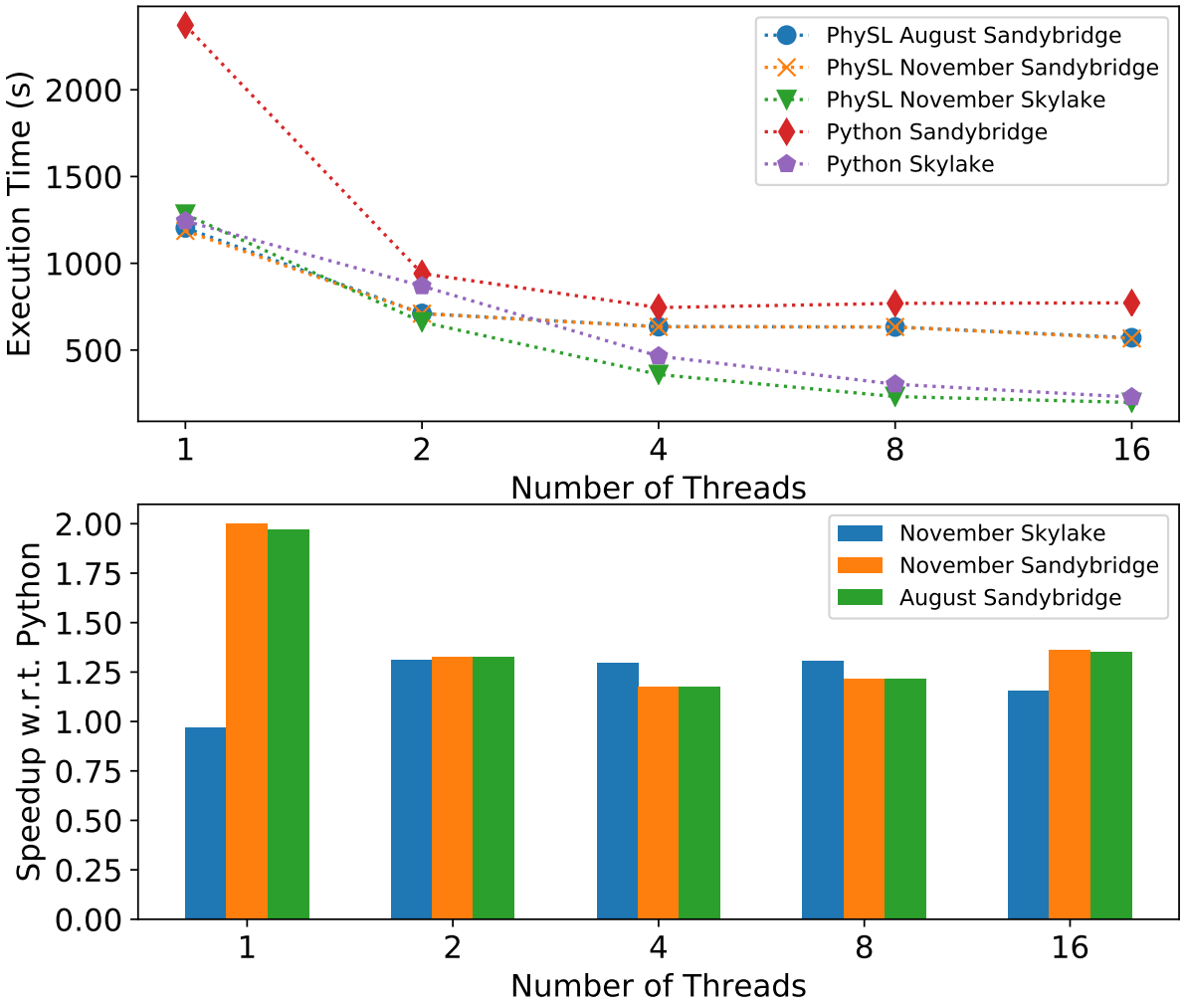

Figure 4: Logistic Regression Performance Comparison with Python

Questions?