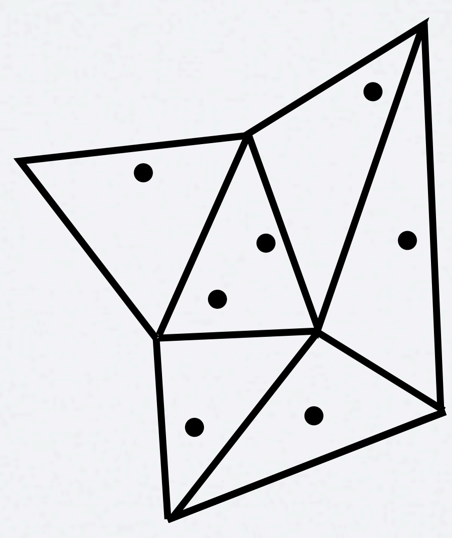
SYNTHESIZING CONCURRENT SCHEDULERS FOR IRREGULAR ALGORITHMS

Donald Nguyen, Keshav Pingali



TERMINOLOGY

irregular algorithm

- opposite: dense arrays
- work on pointered data structures like graphs
- shape of the graph changes during execution
- active nodes + neighborhood
- here: unordered algorithms

TERMINOLOGY

concurrent schedulers

- problem is decomposed into activities
- scheduling assigns them to processors
- order + placement
- static or dynamic scheduling

TERMINOLOGY

static scheduling

- enumerate activities and dependencies at compile-time

dynamic scheduling

- activities are created dynamically
- dependencies cannot be evaluated statically
- execution time estimates unknown

STATE OF THE ART

- dynamic scheduling only for varying execution times
  - OpenMP guided self-scheduling
- irregular algorithms often use **handcrafted schedulers**
  - … even for the sequential implementation
- schedulers are concurrent data structures themselves, so they multiply the parallel programming problem

CONTRIBUTION

Synthesize concrete scheduler implementations from specifications.

ALGORITHMS

- Delaunay Triangulation
- Delaunay Mesh Refinement
- Inclusion-based Points-to Analysis
- Single-source Shortest-path
- Preflow-push

WORKSET

foreach (e : Set S) {B(e)} — The loop body B(e) is executed for each item e of set S. The order in which iterations execute is indeterminate and can be chosen by the implementation. There may be dependences between the iterations. An iteration may add items to S during execution.

SPECIFICATIONS

- global rule for the initial workset
- local rule for thread-local workset
- rules can be composed sequentially

P	$::=$	Global: D Local: D	*Specification*
D	$::=$	R_{NF}^{*} $R_{\mathrm{F}}?$	*Ordering rule*
R_{F}	$::=$	FIFO	*Final rule*
	$\mid$	LIFO	
	$\mid$	Random	
R_{NF}	$::=$	ChunkedFIFO(k)	*Non-final rule*
	$\mid$	ChunkedLIFO(k)	
	$\mid$	Ordered(f_{D})	
	$\mid$	OrderedByMetric(f_{M})	
k			*Integer*
f_{D}			$\mathbf{T} \times \mathbf{T} \rightarrow \mathbf{bool}$
f_{M}			$\mathbf{T} \rightarrow \mathbb{R}$

SPECIFICATIONS

Algorithm	Order	Specification
DMR	Bucketed triangle angle (AS2)	OrderedByMetric($\lambda t.$ minangle(t)) FIFO
	Local stack (AS1)	Global: ChunkedFIFO(k) Local: LIFO
DT	BRIO (AS1)	OrderedByMetric($\lambda p.\ p.round$) ChunkedFIFO(k)
	Random	Random
PFP	FIFO	FIFO
	HL order (AS1)	OrderedByMetric($\lambda n.\ -n.height$) FIFO
PTA	LRF	FIFO
	Split worklists (BS-F)	
SSSP	Bellman-Ford	FIFO
	Delta-stepping (AS1)	OrderedByMetric($\lambda n.\ \lfloor 2 * n.w/\Delta \rfloor + (n.light)$? 0 : 1) FIFO
	Dijkstra (AS2)	Ordered($\lambda a, b.\ a.w \leq b.w$)

IMPLEMENTATION

- workset interface:

```
void add(T t)
T poll()
```

- sequential rule composition by nested worksets
- relaxed poll semantic helps for race-free lock elision

```
T poll-s()
```

OPTIMIZATION

Ignore Size	Use Serial	Use Bounded	$t = 1$	$t = 8$
+	+	+	0.0	0.0
-	+	+	0.8	12.1
+	-	+	2.4	5.5
-	-	+	7.8	7.7
+	+	-	3.6	3.5
-	+	-	11.3	11.5
+	-	-	5.0	16.8
-	-	-	2.9	17.5

GALOIS

- Java
- provides workset iterator
- tracks **active nodes** and **neighborhoods**
- records **undo actions** to **roll back** conflicting activities (overlapping neighborhoods)
- like a coarse-grained STM

EVALUATION

	BASE	RAND	LIFO	FIFO	WS-L	WS-F	BS-L	BS-F	AS2	AS1
					Nehalem					
DMR	12.88	14.80	**11.45**	13.09	11.51	13.27	12.76	13.17	15.56	11.62
DT	25.04					25.42				**14.78**
PFP	110.93	109.77	169.86	115.40	173.47	116.44	110.18	118.59		**45.94**
PTA	13.87	-	-	**12.58**	-	12.74	20.26	12.84		
SSSP	-	-	-	-	-	-	-	-	7.66	**4.96**
					Shanghai					
DMR	16.29	19.52	**13.55**	16.76	13.74	16.76	16.25	16.71	19.59	13.64
DT	43.40					43.55				**27.86**
PFP	237.04	210.57	320.24	237.17	314.53	234.13	216.50	217.67		**74.26**
PTA	19.99	-	-	18.80	-	**18.79**	26.44	18.82		
SSSP	-	-	-	-	-	-	-	-	11.08	**9.53**
					Niagara					
DMR	61.76	68.10	54.79	63.51	**53.84**	63.31	62.86	64.17	77.81	60.33
DT	178.21					179.00				**149.42**
PFP	787.05	734.27	1264.61	741.01	1297.71	775.04	720.20	827.07		**342.41**
PTA	59.17	-	-	57.73	-	57.30	76.16	**56.99**		
SSSP	-	-	-	-	-	-	-	-	33.84	**23.35**

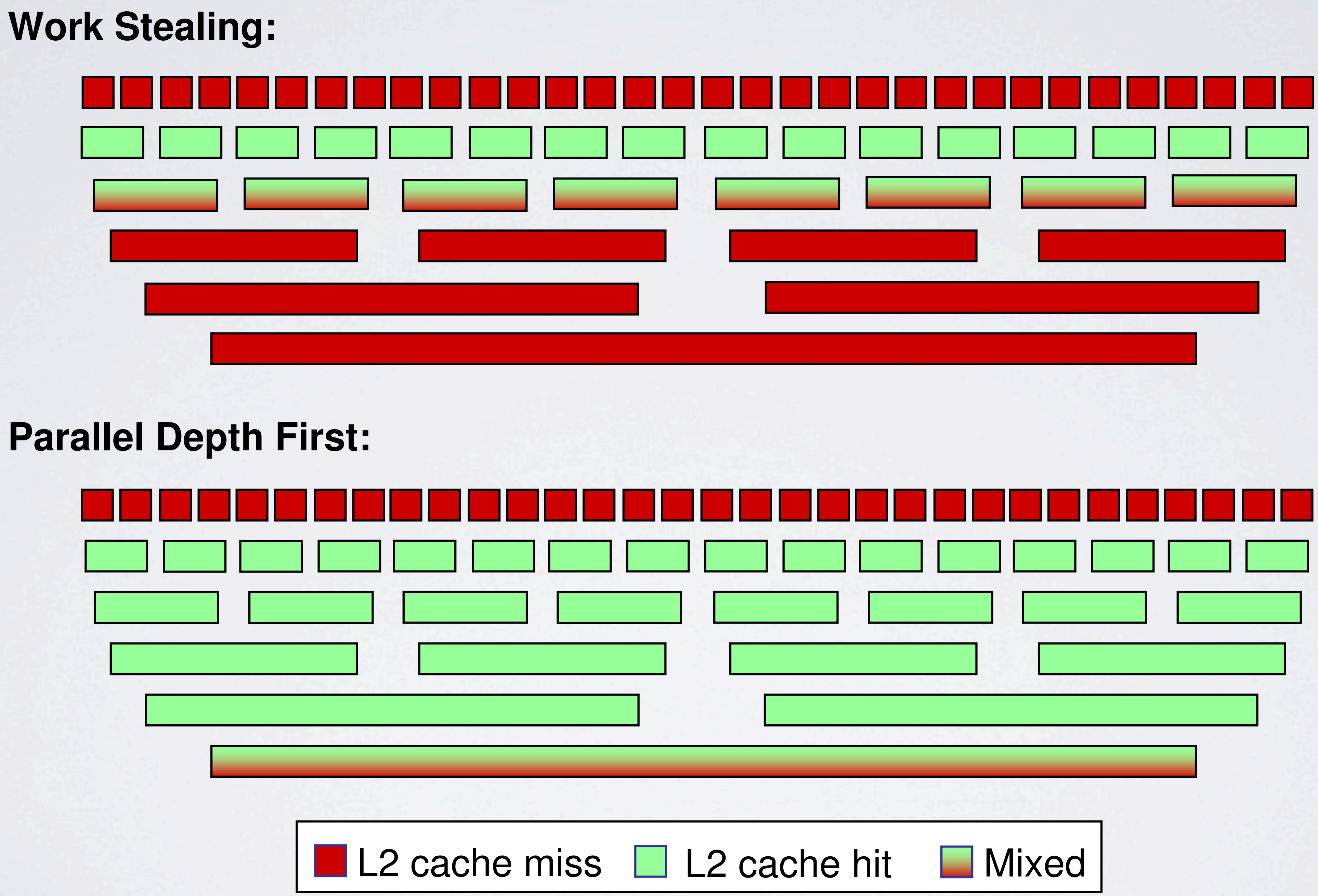
EVALUATION

	BASE	RAND	LIFO	FIFO	WS-L	WS-F	BS-L	BS-F	AS2	AS1
Nehalem ($t \leq 8$)										
DMR	5.70	4.82	0.95	3.81	4.35	5.13	2.64	3.53	2.01	**6.15**
DT	2.21					2.09				**2.35**
PFP	1.30	0.71	0.20	1.15	0.72	2.30	0.37	0.89		**3.35**
PTA	2.83	-	-	3.53	-	2.05	2.37	**3.77**		
SSSP	-	-	-	-	-	-	-	-	0.61	**3.16**
Shanghai ($t \leq 16$)										
DMR	7.85	3.43	0.95	3.74	6.94	7.53	1.91	3.83	2.32	**10.45**
DT	2.64					**2.65**				2.53
PFP	1.28	0.62	0.20	1.00	0.65	2.19	0.37	0.74		**2.56**
PTA	3.69	-	-	3.63	-	3.08	3.25	**5.03**		
SSSP	-	-	-	-	-	-	-	-	0.80	**3.04**
Niagara ($t \leq 32$)										
DMR	18.77	5.95	0.89	6.81	11.47	18.53	3.60	5.89	3.59	**21.53**
DT	5.43					**5.48**				3.29
PFP	2.30	1.25	0.32	2.84	2.18	4.46	0.80	2.13		**5.92**
PTA	4.20	-	-	4.49	-	5.42	4.62	**6.16**		
SSSP	-	-	-	-	-	-	-	-	0.50	**2.33**

SCHEDULING THREADS FOR CONSTRUCTIVE CACHE SHARING ON CMPS

Shimin Chen et al.
CMU & Intel Pittsburgh

MERGESORT



Work Stealing:
Parallel Depth First:
L2 cache miss
L2 cache hit
Mixed

DISCUSSION

- nice lightweight specification language for workset iterator
  - brings research on parallel algorithms & data structures closer to programming reality
- looks amenable to a C++ template implementation
  - I would like to see a non-Java evaluation
  - maybe someone should repeat the measurements…

GCD

`dispatch_queue_create`

Creates a new dispatch queue to which blocks can be submitted.

```
dispatch_queue_t dispatch_queue_create(
  const char *label,
  dispatch_queue_attr_t attr);
```

attr Currently unused; pass NULL.