Modeling Energy Consumption of Lock-Free Queue Implementations

Aras Atalar, Anders Gidenstam,
Paul Renaud-Goud and Philippas Tsigas

Chalmers University of Technology

Outline

- Motivation and Setting
- Enqueue/Dequeue Throughput Estimation
- Power Estimation
- Results

Introduction

Motivation

Modeling Energy Consumption of Lock-Free Queue Implementations

Motivation

Modeling Energy Consumption of Lock-Free Queue Implementations

- FIFO (First In, First Out) queues:
  - Key components in applications, algorithms, run-time and operating systems
  - Producer/consumer pattern: common approach to parallelizing applications

Motivation

Modeling Energy Consumption of Lock-Free Queue Implementations

- FIFO (First In, First Out) queues:
  - Key components in applications, algorithms, run-time and operating systems
  - Producer/consumer pattern: common approach to parallelizing applications
- Lock-freedom:
  - High concurrency
  - Immunity to deadlocks and convoying

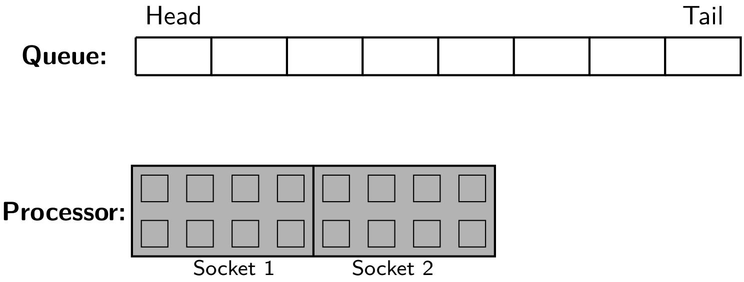
Modeling Energy Consumption of Lock-Free Queue Implementations

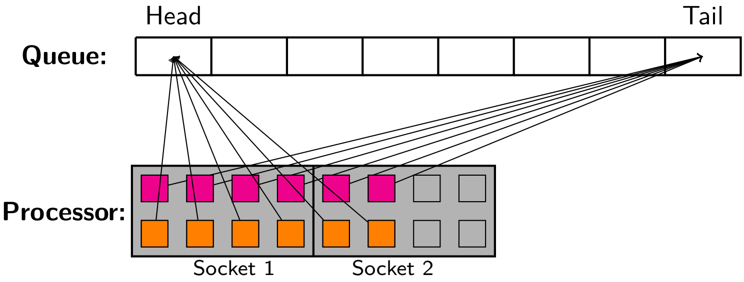
- FIFO (First In, First Out) queues:
  - Key components in applications, algorithms, run-time and operating systems
  - Producer/consumer pattern: common approach to parallelizing applications
- Lock-freedom:
  - High concurrency
  - Immunity to deadlocks and convoying
- Major optimization criterion (Exascale, battery lifetime, *etc.*). Decomposed into:
  - Power
  - Throughput

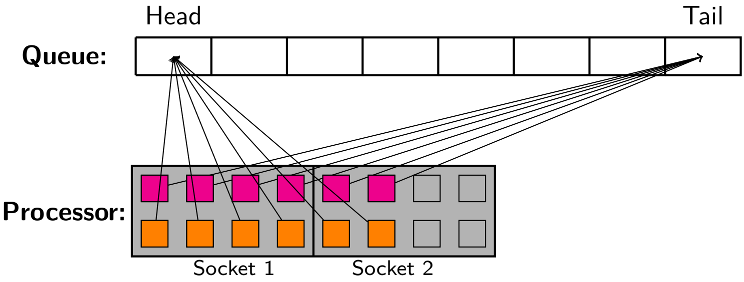
Motivation

Modeling Energy Consumption of Lock-Free Queue Implementations

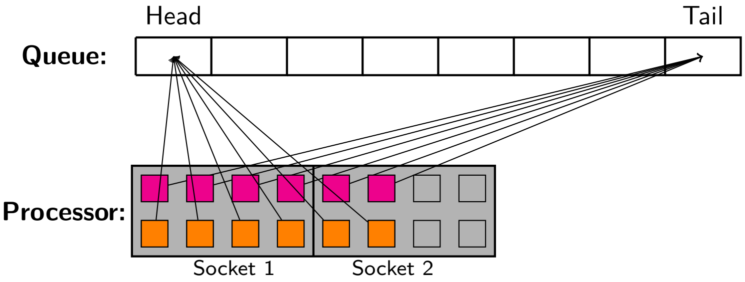
- FIFO (First In, First Out) queues:
  - Key components in applications, algorithms, run-time and operating systems
  - Producer/consumer pattern: common approach to parallelizing applications
- Lock-freedom:
  - High concurrency
  - Immunity to deadlocks and convoying
- Major optimization criterion (Exascale, battery lifetime, *etc.*). Decomposed into:
  - Power
  - Throughput
- Large number of lock-free (and wait-free) queue implementations in the literature
  $\rightsquigarrow$ need of a framework to rank the different implementations, according to throughput, power, energy per operation


Head
Tail
Queue:
Processor:
Socket 1
Socket 2


Head
Tail
Queue:
Processor:
Socket 1
Socket 2



- Different though **constant** access rates for enqueuers and dequeuers
- Steady-state: queue either mostly empty or constantly growing



- Different though **constant** access rates for enqueuers and dequeuers
- Steady-state: queue either mostly empty or constantly growing
- Domain of study:
  (i) nb. of threads accessing the queue
  (ii) CPU frequencies
  (iii) range of dequeue access rates
  (iv) range of enqueue access rates

Algorithm Skeletons

```
while ! done do
    el ← Parallel_Work(pw_e);
    Enqueue(el);
end
```

Procedure EnqueuerThread

```
while ! done do
    el ← Dequeue();
    Parallel_Work(pw_d);
end
```

Procedure DequeuerThread

- Parallel sections (Parallel_Work): processing activity implemented by sequences of bunches of *pause* instructions in the benchmark
- Enqueue and Dequeue: retry loop pattern

```
repeat
    Try to Enqueue
until Succesful;
```

Procedure Enqueue

```
repeat
    Try to Dequeue
until Succesful;
```

Procedure Dequeue

Notations

- n: number of threads that call the same type of operation
- f: clock frequency

Notations

- n: number of threads that call the same type of operation
- f: clock frequency

- $\left.\begin{array}{l} pw_e \\ pw_d \end{array}\right\}$ amount of work in the parallel section of $\left\{\begin{array}{l} \text{an enqueuer} \\ \text{a dequeuer} \end{array}\right.$
- $\left.\begin{array}{l} rw_e \\ rw_d \end{array}\right\}$ amount of work in one retry of the retry loop of $\left\{\begin{array}{l} \text{Enqueue} \\ \text{Dequeue} \end{array}\right.$

Notations

- n: number of threads that call the same type of operation
- f: clock frequency

- $\left.\begin{array}{l} pw_e \\ pw_d \end{array}\right\}$ amount of work in the parallel section of $\left\{\begin{array}{l} \text{an enqueuer} \\ \text{a dequeuer} \end{array}\right.$
- $\left.\begin{array}{l} rw_e \\ rw_d \end{array}\right\}$ amount of work in one retry of the retry loop of $\left\{\begin{array}{l} \text{Enqueue} \\ \text{Dequeue} \end{array}\right.$

- $\mathcal{T}_e$: throughput of enqueuers
- $\mathcal{T}_d$: throughput of dequeuers

Notations

- n: number of threads that call the same type of operation
- f: clock frequency
- $\left.\begin{matrix} pw_e \\ pw_d \end{matrix}\right\}$ amount of work in the parallel section of $\left\{\begin{matrix} \text{an enqueuer} \\ \text{a dequeuer} \end{matrix}\right.$
- $\left.\begin{matrix} rw_e \\ rw_d \end{matrix}\right\}$ amount of work in one retry of the retry loop of $\left\{\begin{matrix} \text{Enqueue} \\ \text{Dequeue} \end{matrix}\right.$
- $\mathcal{T}_e$: throughput of enqueuers
- $\mathcal{T}_d$: throughput of dequeuers
- For operation $o \in \{d, e\}$:

$$\mathcal{T}_o = \frac{n \times f}{pw_o + rw_o \times Repeat_o}$$

Throughput Estimation

- Operations behavior depends on the state of the queue (empty or not empty).
- Contention

Impacting factors

- Operations behavior depends on the state of the queue (empty or not empty).
- Contention is twofold:
  - intra-contention: competition between threads executing the same operation
  - inter-contention: competition between threads executing different operations. Occurs when mostly empty queue.

Impacting factors

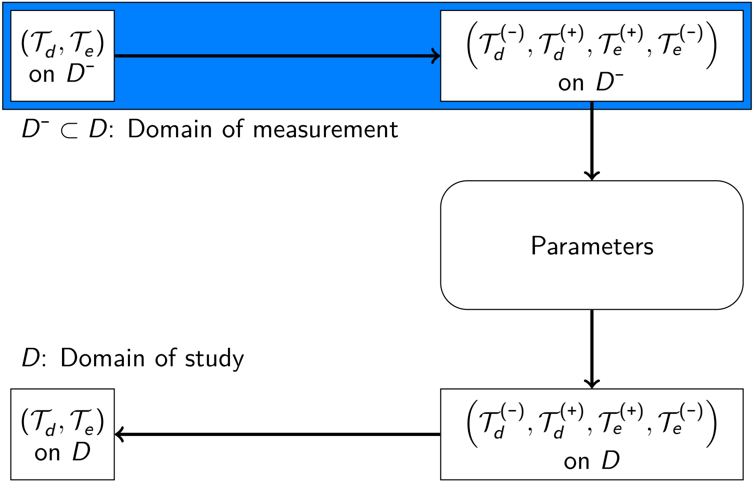
- Operations behavior depends on the state of the queue (empty or not empty).
- Contention is twofold:
  - intra-contention: competition between threads executing the same operation
  - inter-contention: competition between threads executing different operations. Occurs when mostly empty queue.
- Impact of state of the queue on Enqueue negligible: same instructions.
- Impact of inter-contention on Dequeue negligible: only a single Enqueuer can interfere, since after a success, the queue is not empty.

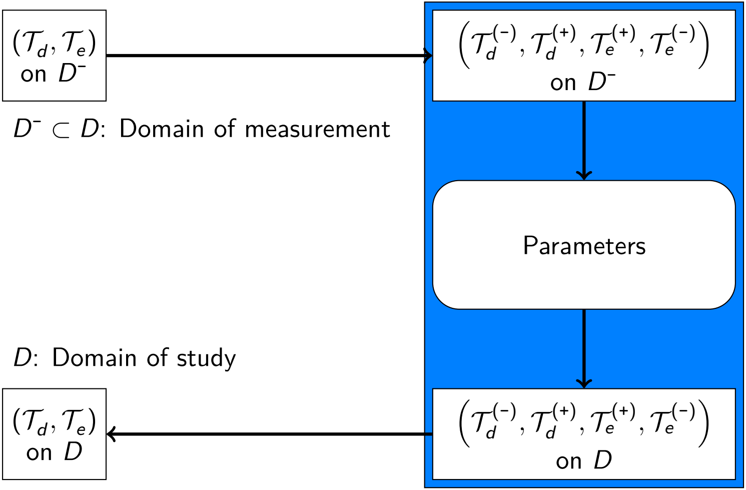
Highlight Impacting Factors

- For operation $o \in \{d, e\}$:

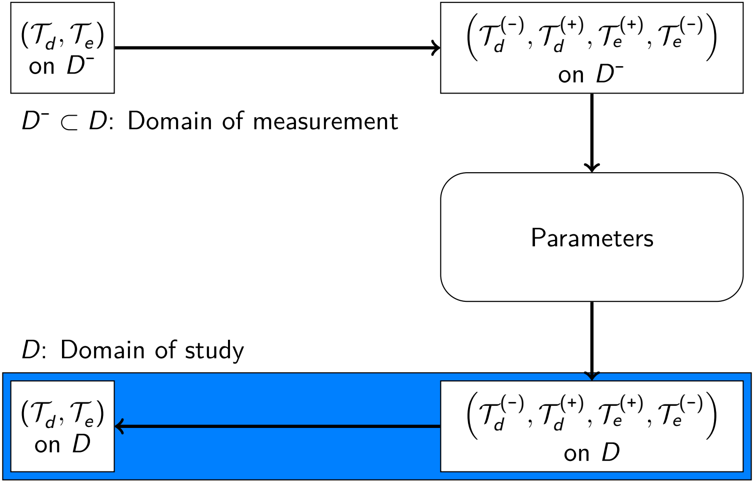
$$\mathcal{T}_o = \frac{n \times f}{pw_o + rw_o \times Repeat_o}$$

- Intra-contention:
  - $Repeat_o$ increases due to interferences
  - rw_d, rw_e increases/expands due to serialization of atomic instructions
- Inter-operation effects:
  - Inter-contention: rw_e increases
  - State of the queue: rw_d variates between NULL and not NULL cost


$(\mathcal{T}_d, \mathcal{T}_e)$ on D^-
$\left(\mathcal{T}_d^{(-)}, \mathcal{T}_d^{(+)}, \mathcal{T}_e^{(+)}, \mathcal{T}_e^{(-)}\right)$ on D^-
$D^- \subset D$: Domain of measurement
Parameters
D: Domain of study
$(\mathcal{T}_d, \mathcal{T}_e)$ on D
$\left(\mathcal{T}_d^{(-)}, \mathcal{T}_d^{(+)}, \mathcal{T}_e^{(+)}, \mathcal{T}_e^{(-)}\right)$ on D


$(\mathcal{T}_d, \mathcal{T}_e)$ on D^-
$\left(\mathcal{T}_d^{(-)}, \mathcal{T}_d^{(+)}, \mathcal{T}_e^{(+)}, \mathcal{T}_e^{(-)}\right)$ on D^-
$D^- \subset D$: Domain of measurement
Parameters
D: Domain of study
$(\mathcal{T}_d, \mathcal{T}_e)$ on D
$\left(\mathcal{T}_d^{(-)}, \mathcal{T}_d^{(+)}, \mathcal{T}_e^{(+)}, \mathcal{T}_e^{(-)}\right)$ on D

Throughput Estimation



$$\mathcal{T} = \min \begin{pmatrix} \mathcal{T}_e \\ , \\ \mathcal{T}_d \end{pmatrix}$$

- $\mathcal{T}_e \to \mathcal{T}_e^{(+)}$: under minimum inter-contention
- $\mathcal{T}_e \to \mathcal{T}_e^{(-)}$: under maximum inter-contention
- $\mathcal{T}_d \to \mathcal{T}_d^{(+)}$: Dequeue NULL element
- $\mathcal{T}_d \to \mathcal{T}_d^{(-)}$: Dequeue not NULL element

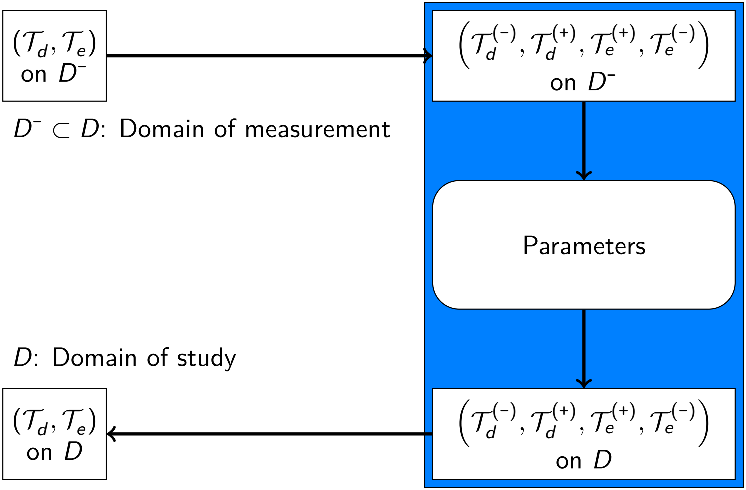
For operation $o \in \{d, e\}$, $\mathcal{T}_o$ barycenter between $\mathcal{T}_o^{(+)}$ and $\mathcal{T}_o^{(-)}$

$$\begin{cases} \mathcal{T}_e(pw_d, pw_e) = (1 - \alpha_e(pw_d, pw_e))\mathcal{T}_e^{(+)}(pw_e) + \alpha_e(pw_d, pw_e)\mathcal{T}_e^{(-)}(pw_e) \\ \mathcal{T}_d(pw_d, pw_e) = (1 - \alpha_d(pw_d, pw_e))\mathcal{T}_d^{(+)}(pw_d) + \alpha_d(pw_d, pw_e)\mathcal{T}_d^{(-)}(pw_d) \end{cases}$$

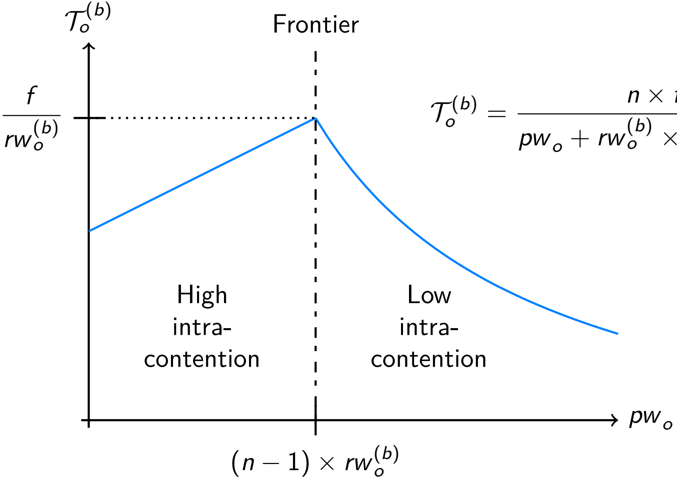
$\rightsquigarrow$ decorrelation of dependencies.

Expressions of the four basic throughputs $\mathcal{T}_o^{(b)}$, for $o \in \{e, d\}$ and $b \in \{+, -\}$, and weights α_o, for $o \in \{e, d\}$?

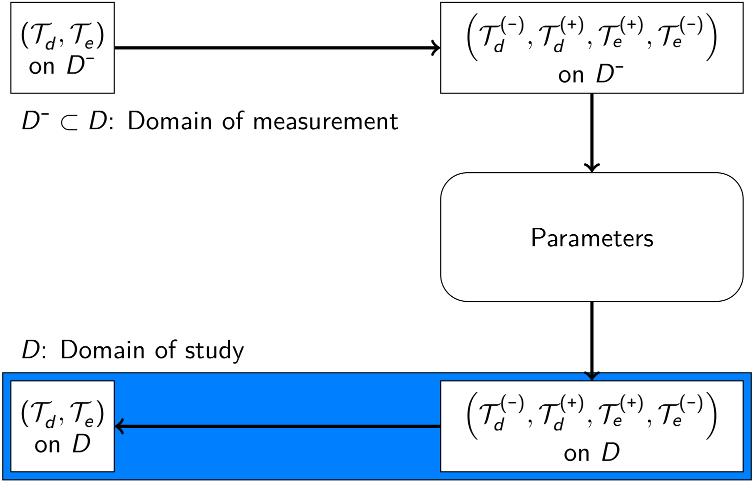
Throughput Estimation



Handle Intra-Contention



$$\mathcal{T}_o^{(b)} = \frac{n \times f}{pw_o + rw_o^{(b)} \times Repeat_o^{(b)}}$$


$(\mathcal{T}_d, \mathcal{T}_e)$ on D^-
$\left(\mathcal{T}_d^{(-)}, \mathcal{T}_d^{(+)}, \mathcal{T}_e^{(+)}, \mathcal{T}_e^{(-)}\right)$ on D^-
$D^- \subset D$: Domain of measurement
Parameters
D: Domain of study
$(\mathcal{T}_d, \mathcal{T}_e)$ on D
$\left(\mathcal{T}_d^{(-)}, \mathcal{T}_d^{(+)}, \mathcal{T}_e^{(+)}, \mathcal{T}_e^{(-)}\right)$ on D

Combining throughputs

Combination is based on the two possible states of the queue:

- Mostly empty
- Growing

Combination is based on the two possible states of the queue:

- Mostly empty
- Growing

If the queue is growing,

- No inter-contention
- All dequeued element is not NULL

Combining throughputs

Combination is based on the two possible states of the queue:

- Mostly empty
- Growing

If the queue is growing,

- No inter-contention $\rightsquigarrow \mathcal{T}_e = \mathcal{T}_e^{(+)}$
- All dequeued element is not NULL $\rightsquigarrow \mathcal{T}_d = \mathcal{T}_d^{(-)}$

Combining throughputs

Combination is based on the two possible states of the queue:

- Mostly empty
- Growing

If the queue is growing,

- No inter-contention $\rightsquigarrow \mathcal{T}_e = \mathcal{T}_e^{(+)}$
- All dequeued element is not $\textsc{Null} \rightsquigarrow \mathcal{T}_d = \mathcal{T}_d^{(-)}$

If the queue is mostly empty,

- Inter-contention occurs
- Both $\textsc{Null}$ and not $\textsc{Null}$ elements are dequeued

$\rightsquigarrow \mathcal{T}_e$ between $\mathcal{T}_e^{(+)}$ and $\mathcal{T}_e^{(-)}$, and $\mathcal{T}_d$ between $\mathcal{T}_d^{(-)}$ and $\mathcal{T}_d^{(+)}$.

Power Estimation

Power split into:

- *Static* part: cost of turning the machine on
- *Activation* part: fixed cost for each socket and each core in use
- *Dynamic* part: supplementary cost depending on the running application

In accordance with the RAPL energy counters, each part further decomposed per-component:

- Memory
- CPU
- *Uncore*

Finally,

$$P = \sum_{X \in \{M,C,U\}} \left(P^{(stat,X)} + P^{(active,X)} + P^{(dyn,X)} \right)$$

- Dynamic memory power is proportional to the intensity (number of units of memory accessed per unit of time) of main memory accesses and inter-socket communication
- Communications only in the retry loop
- Assumption: for a given implementation, constant intensity in the retry loop
- $\rightsquigarrow$ Dynamic memory power dissipated in the retry loop proportional to r_o (times a constant intensity)

$$P^{(M)} = r_e \times \rho_e^{(M)} + r_d \times \rho_d^{(M)},$$

where $\rho_e^{(M)}$ and $\rho_d^{(M)}$ are constants

- Dynamic memory power is proportional to the intensity (number of units of memory accessed per unit of time) of main memory accesses and inter-socket communication
- Communications only in the retry loop
- Assumption: for a given implementation, constant intensity in the retry loop
- $\rightsquigarrow$ Dynamic memory power dissipated in the retry loop proportional to r_o (times a constant intensity)

$$P^{(M)} = r_e \times \rho_e^{(M)} + r_d \times \rho_d^{(M)},$$

  where $\rho_e^{(M)}$ and $\rho_d^{(M)}$ are constants
- Uncore and CPU power computed with similar principles

Results

Queue implementations

[1] Maged M. Michael and Michael L. Scott. "Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms". In: *PoDC*. 1996, pp. 267–275.

[2] J. D. Valois. "Implementing Lock-Free Queues". In: *ICPADS*. 1994, pp. 64–69.

[3] Philippas Tsigas and Yi Zhang. "A Simple, Fast and Scalable Non-Blocking Concurrent FIFO queue for Shared Memory Multiprocessor Systems". In: *SPAA*. 2001, pp. 134–143.

[4] Anders Gidenstam, Håkan Sundell, and Philippas Tsigas. "Cache-Aware Lock-Free Queues for Multiple Producers/Consumers and Weak Memory Consistency". In: *OPODIS*. Vol. 6490. 2010, pp. 302–317.

[5] Moshe Hoffman, Ori Shalev, and Nir Shavit. "The Baskets Queue". In: *OPODIS*. 2007, pp. 401–414.

[6] Mark Moir, Daniel Nussbaum, Ori Shalev, and Nir Shavit. "Using elimination to implement scalable and lock-free FIFO queues". In: *SPAA*. 2005, pp. 253–262. ISBN: 1-58113-986-1. DOI: http://doi.acm.org/10.1145/1073970.1074013.

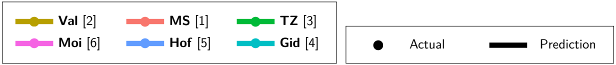
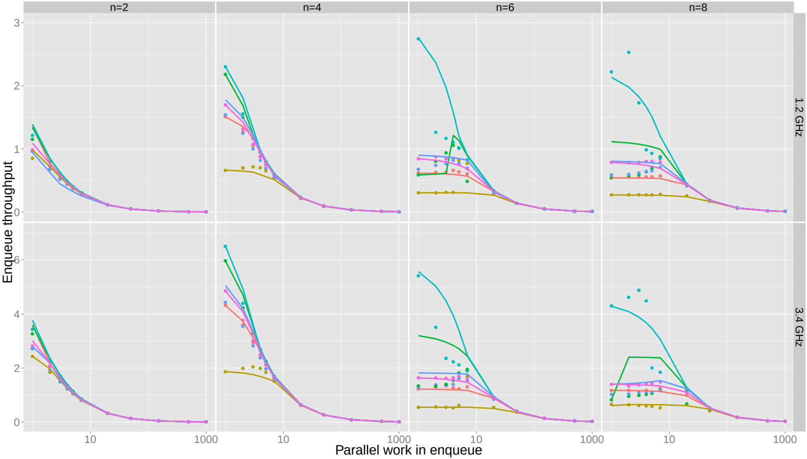
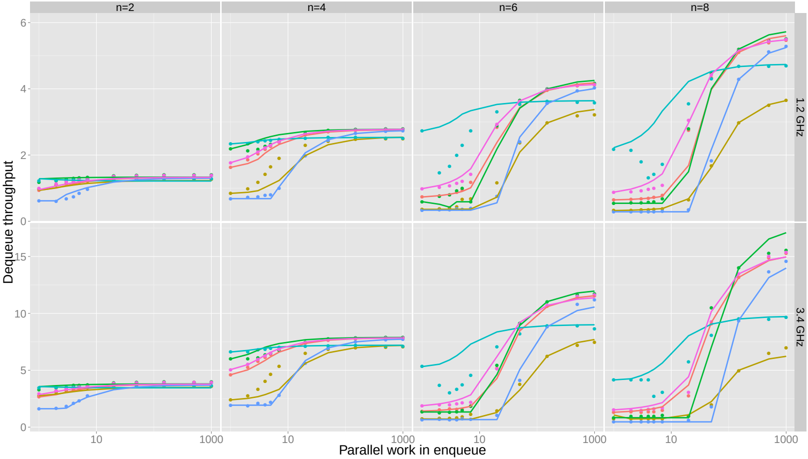


Figure : Key legend of the graphs

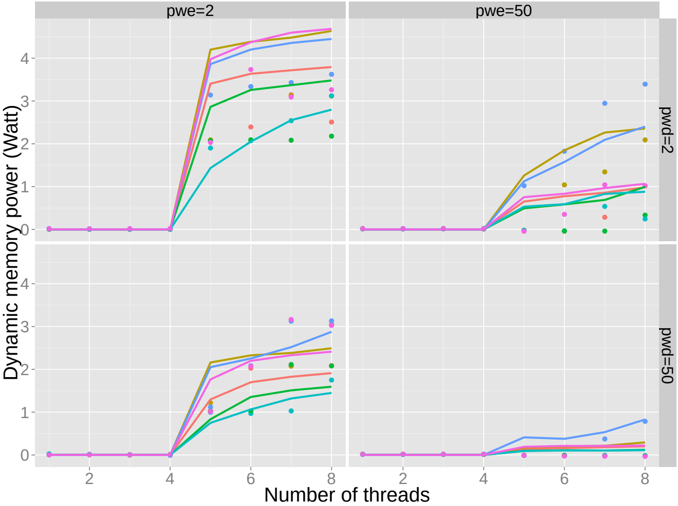
$$pw_d = 1$$


n=2
n=4
n=6
n=8
1.2 GHz
3.4 GHz
Enqueue throughput
Parallel work in enqueue
0
1
2
3
4
6
10
1000

$$pw_d = 1$$


n=2
n=4
n=6
n=8
1.2 GHz
3.4 GHz
Dequeue throughput
Parallel work in enqueue
0
2
4
6
5
10
15
10
1000

Dynamic Memory Power



Dynamic memory power at $f = 3.4\,\text{GHz}$

- Model of power, throughput and energy per operation of lock-free queues
- Validation on several widely-used implementations
- Decomposition into basic throughputs thanks to two impacting factors
  - Inter- and intra-contention
  - State of the queue
- $\rightsquigarrow$ better understanding and reduction of the number of measurement points
- Generalization to slowly changing parallel sections on Mandelbrot application