CPSC 768:
Scalable and Private Graph Algorithms

Lecture 23: Algorithms Engineering: Theory-in-Practice

Quanquan C. Liu
quanquan.liu@yale.edu

Announcements

- **Final project report and presentation: April 24th (last day of class)**
  - Final project presentation is a 30 min presentation
- **Last day of Open Problem Sessions: April 26th** (last week of classes)
  - Will be turned into a reading group/continue with OPS, stay tuned!

Continued: Parallel Low Diameter Decomposition

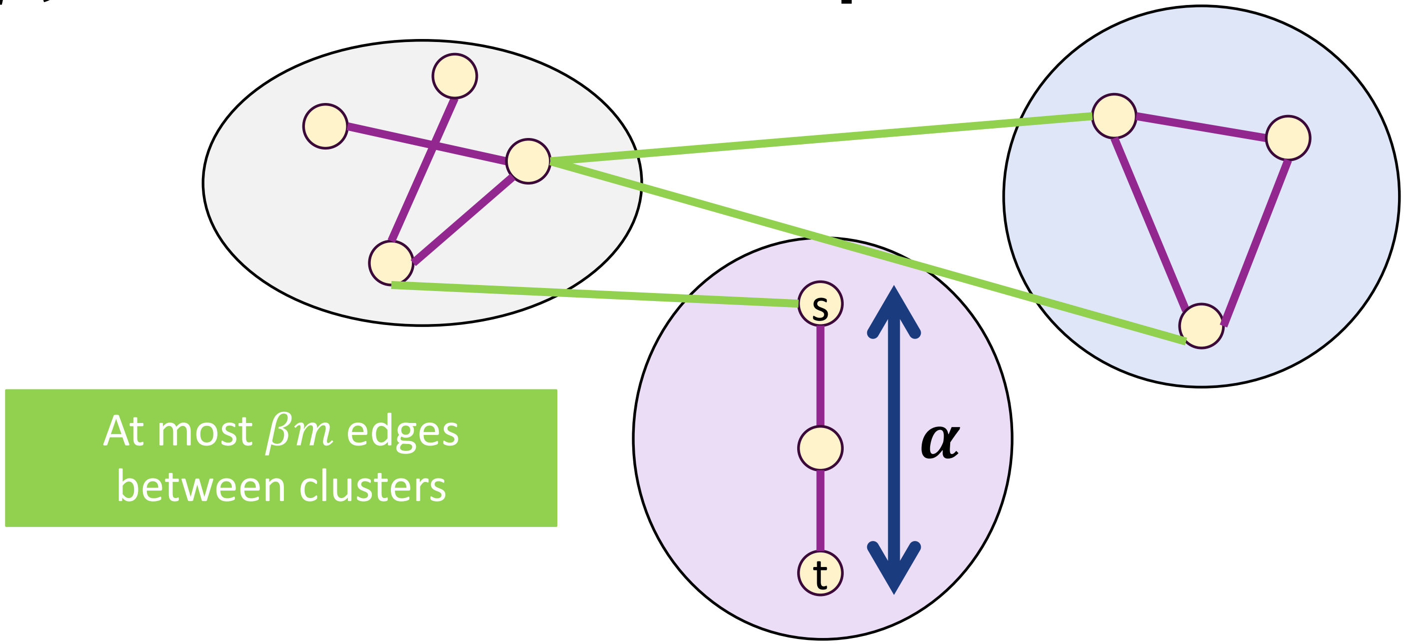
- **Problem:** Decompose the graph into clusters where the distance between any two nodes in a cluster is at most α and the number of edges that go between clusters is at most βm

Continued: Parallel Low Diameter Decomposition

- **Problem:** Decompose the graph into clusters where the distance between any two nodes in a cluster is at most α and the number of edges that go between clusters is at most βm
  - $(\boldsymbol{\alpha}, \boldsymbol{\beta})$**- low diameter decomposition**

Continued: Parallel Low Diameter Decomposition

- **Problem:** Decompose the graph into clusters where the distance between any two nodes in a cluster is at most α and the number of edges that go between clusters is at most βm
  - $(\boldsymbol{\alpha}, \boldsymbol{\beta})$**- low diameter decomposition**



Parallel Low Diameter Decomposition [MPX SPAA ‘13]

- **Algorithm:**
  - Grow balls around carefully selected vertices in the input graph
  - How do we select the vertices?

Parallel Low Diameter Decomposition [MPX SPAA ‘13]

- **Algorithm:**
  - Grow balls around carefully selected vertices in the input graph
  - How do we select the vertices?
    - From **exponential distribution** with parameter β

Parallel Low Diameter Decomposition [MPX SPAA ‘13]

- **Algorithm:**
  - Grow balls around carefully selected vertices in the input graph
  - How do we select the vertices?
    - From **exponential distribution** with parameter β
    - Start growing from vertices where start times are randomly shifted by exponential distribution

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Algorithm:**
  - Grow balls around carefully selected vertices in the input graph
  - How do we select the vertices?
    - From **exponential distribution** with parameter β
    - Start growing from vertices where start times are randomly shifted by exponential distribution
    - Expand the radius using BFS

Parallel Low Diameter Decomposition [MPX SPAA ‘13]

- **Algorithm:**
  - Grow balls around carefully selected vertices in the input graph
  - How do we select the vertices?
    - From **exponential distribution** with parameter β
    - Start growing from vertices where start times are randomly shifted by exponential distribution
    - Expand the radius using BFS
    - Vertex assigned to first edge that hits it

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **PDF of exponential distribution:**
  - $p(x) = \beta e^{-\beta x}$
- **CDF of exponential distribution:**
  - $c(x) = 1 - e^{-\beta x}$

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **PDF of exponential distribution:**
  - $p(x) = \beta e^{-\beta x}$
- **CDF of exponential distribution:**
  - $c(x) = 1 - e^{-\beta x}$
- **Exponential Function is memoryless:**
  - $P[X > a + b \mid X > a] = P[X > b]$

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **PDF of exponential distribution:**
  - $p(x) = \beta e^{-\beta x}$
- **CDF of exponential distribution:**
  - $c(x) = 1 - e^{-\beta x}$
- **Exponential Function is memoryless:**
  - $P[X > a + b \mid X > a] = P[X > b]$
- Each vertex v picks time δ_v from exponential distribution

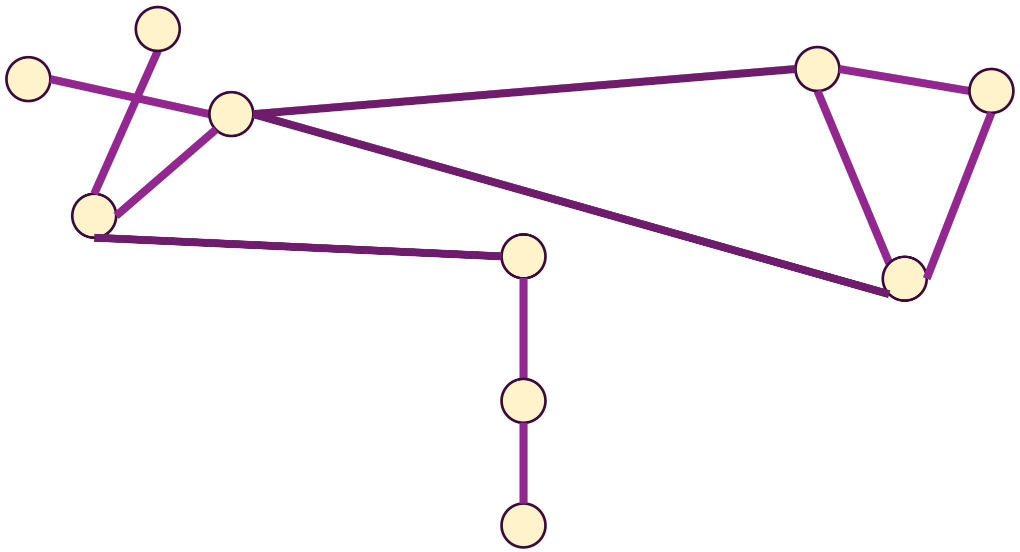
Parallel Low Diameter Decomposition [MPX SPAA '13]

- **PDF of exponential distribution:**
  - $p(x) = \beta e^{-\beta x}$
- **CDF of exponential distribution:**
  - $c(x) = 1 - e^{-\beta x}$
- **Exponential Function is memoryless:**
  - $P[X > a + b \mid X > a] = P[X > b]$
- Each vertex v picks time δ_v from exponential distribution
- Use start time $T_v = \delta_{\max} - \delta_v$ where $\delta_{\max} = \max_{v \in V}(\delta_v)$

Backwards Exponential Distribution: very few to start, more towards the end

Parallel Low Diameter Decomposition [MPX SPAA '13]

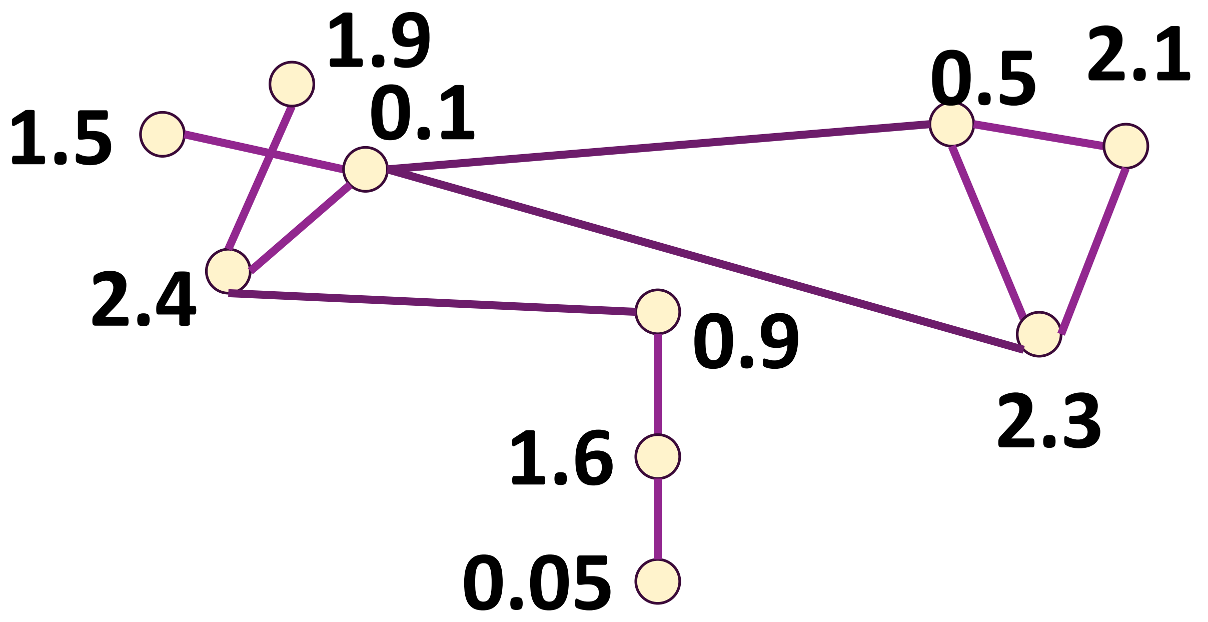
- **Example Run:**



First: everyone picks value from exponential distribution

Parallel Low Diameter Decomposition [MPX SPAA '13]

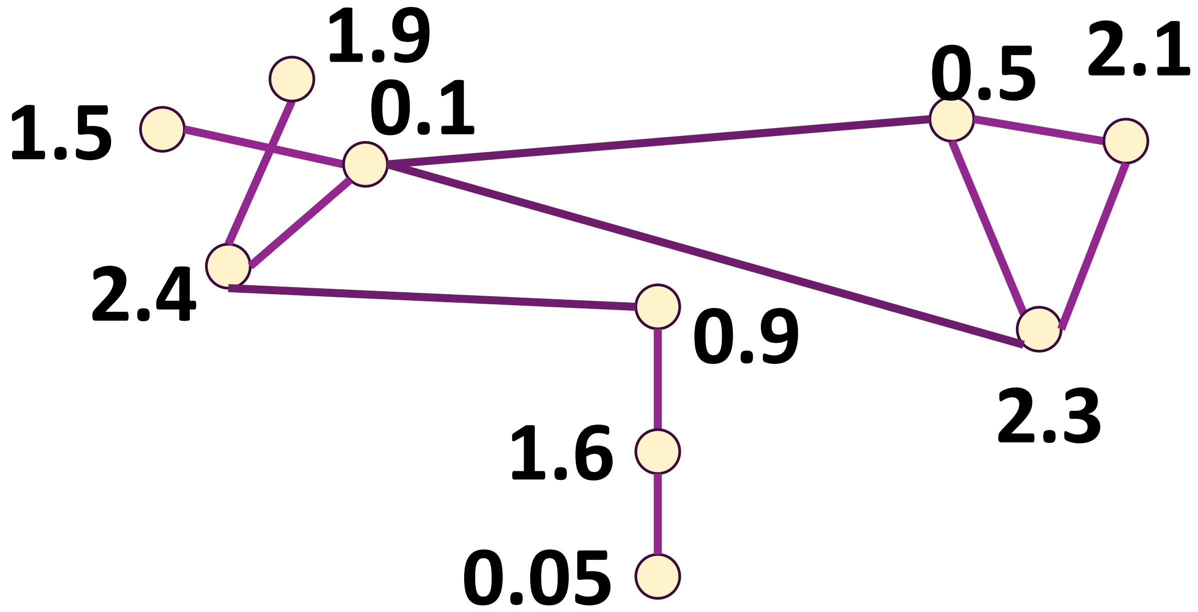
- **Example Run:**



First: everyone picks value from exponential distribution

Parallel Low Diameter Decomposition [MPX SPAA '13]

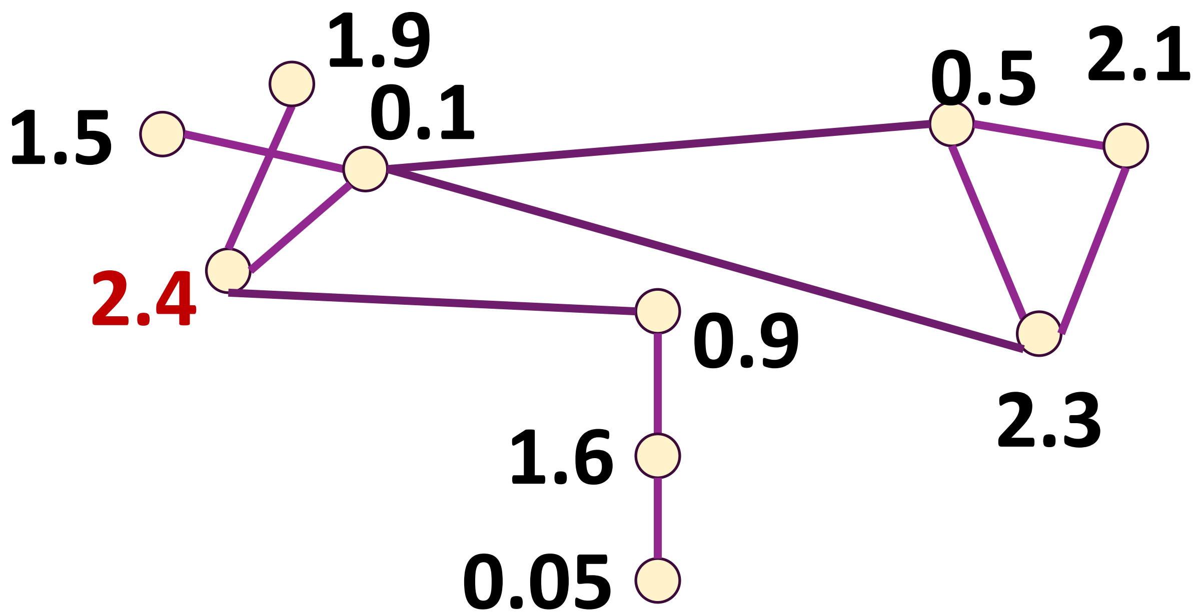
- **Example Run:**



Find $\delta_{\max}$ and compute T_v for every vertex

Parallel Low Diameter Decomposition [MPX SPAA '13]

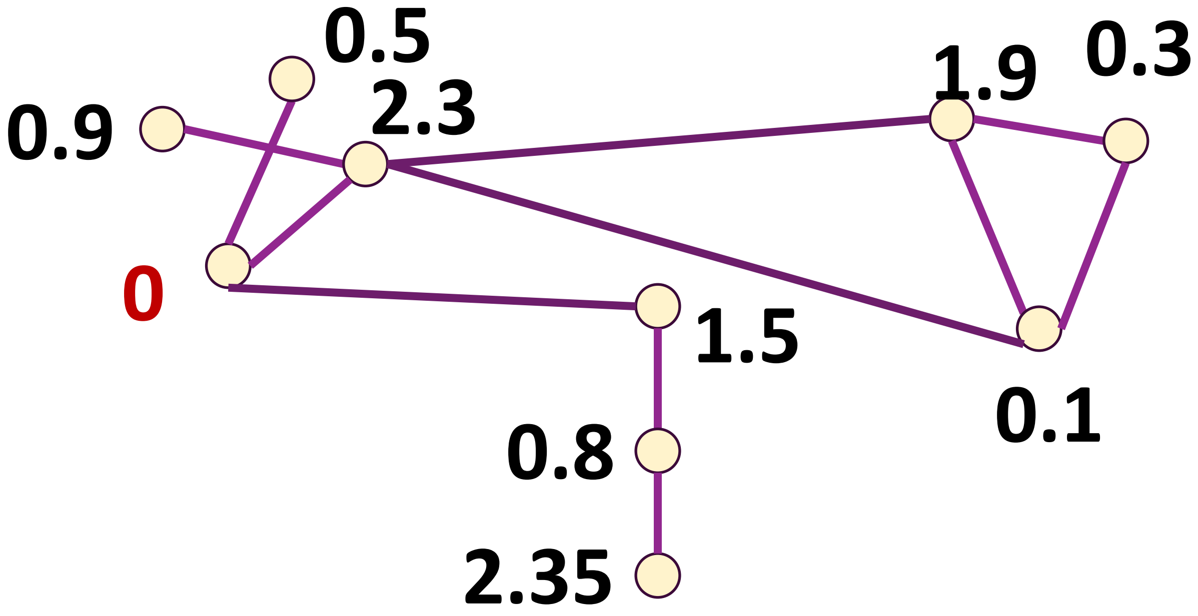
- **Example Run:**



Find $\delta_{\max}$ and compute T_v for every vertex

Parallel Low Diameter Decomposition [MPX SPAA '13]

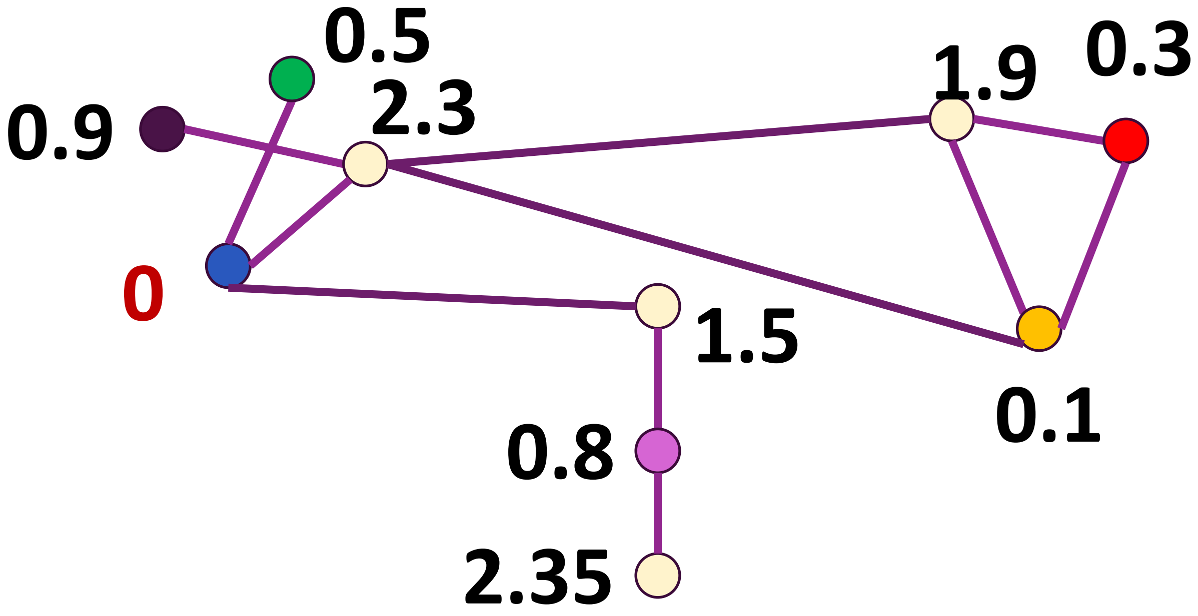
- **Example Run:**



Find $\delta_{\max}$ and compute T_v for every vertex

Parallel Low Diameter Decomposition [MPX SPAA '13]

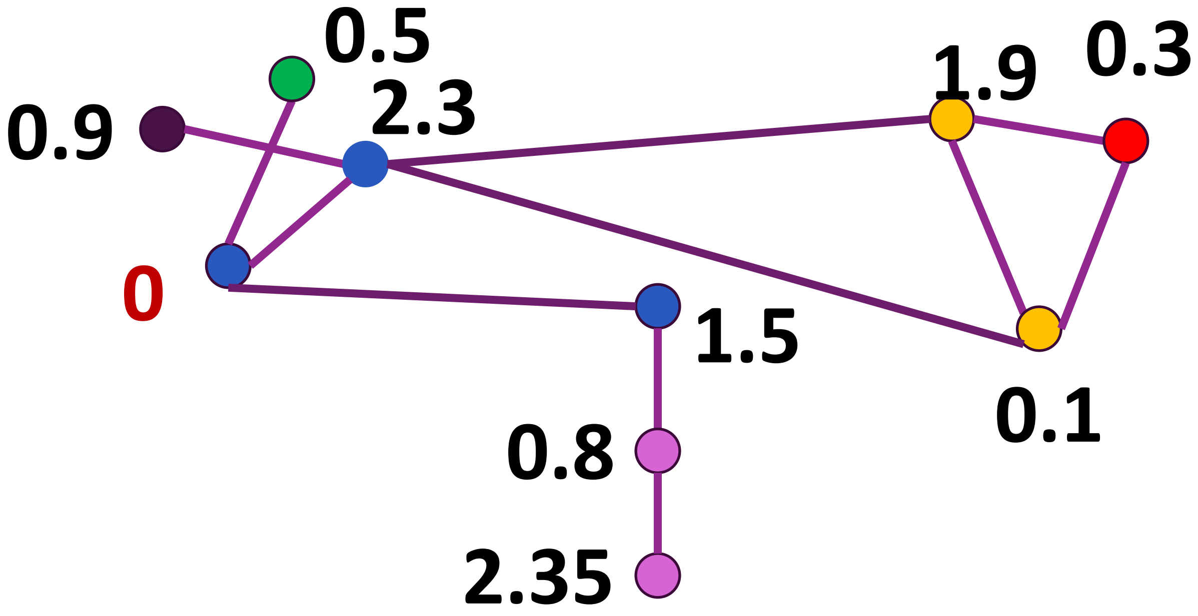
- **Example Run:**



Every timestep, grow radius by 1

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Example Run:**



Every timestep, grow radius by 1

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Analysis of Low Diameter:**
  - Let $C_u = \underset{u \in V}{\operatorname{argmin}}\left(T_v + d(u,v)\right)$
  - Maximum radius is $\boldsymbol{O\left(\frac{\log(n)}{\beta}\right)}$ **with high probability**

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Analysis of Low Diameter:**
  - Let $C_u = \underset{u \in V}{\operatorname{argmin}} \left(T_v + d(u, v) \right)$
  - Maximum radius is $\boldsymbol{O\left(\frac{\log(n)}{\beta}\right)}$ **with high probability**
  - **What is the radius bounded by?**

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Analysis of Low Diameter:**
  - Let $C_u = \operatorname*{argmin}_{u \in V}\left(T_v + d(u, v)\right)$
  - Maximum radius is $\boldsymbol{O\left(\frac{\log(n)}{\beta}\right)}$ **with high probability**
  - **What is the radius bounded by?**
    - **All radius bounded by $\boldsymbol{\delta_{\max}}$**

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Analysis of Low Diameter:**
  - Let $C_u = \underset{u \in V}{\operatorname{argmin}}\left(T_v + d(u, v)\right)$
  - Maximum radius is $\boldsymbol{O}\left(\frac{\mathbf{log}(\boldsymbol{n})}{\boldsymbol{\beta}}\right)$ **with high probability**
  - **What is the radius bounded by?**
    - **All radius bounded by $\boldsymbol{\delta}_{\mathbf{max}}$**
  - Simplified: just need to bound $\delta_{\max}$ using CDF of exponential distribution

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Analysis of Low Diameter:**
  - Let $C_u = \operatorname*{argmin}_{u \in V}\left(T_v + d(u,v)\right)$
  - Maximum radius is $\boldsymbol{O\left(\frac{\log(n)}{\beta}\right)}$ **with high probability**
  - **What is the radius bounded by?**
    - **All radius bounded by $\boldsymbol{\delta_{\max}}$**
  - Simplified: just need to bound $\delta_{\max}$ using CDF of exponential distribution
    - $\boxed{P\left[\delta_v > \frac{c \log n}{\beta}\right] = 1 - P\left[\delta_v \le \frac{c \log n}{\beta}\right] = 1 - \left(1 - e^{-c \log n}\right) = \frac{1}{n^c}}$

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Analysis of Low Diameter:**
  - Let $C_u = \underset{u \in V}{\operatorname{argmin}}\left(T_v + d(u, v)\right)$
  - Maximum radius is $\boldsymbol{O}\left(\frac{\log(n)}{\beta}\right)$ **with high probability**
  - **What is the radius bounded by?**
    - **All radius bounded by $\delta_{\max}$**
  - Simplified: just need to bound $\delta_{\max}$ using CDF of exponential distribution
    - $P\left[\delta_v > \frac{c \log n}{\beta}\right] = 1 - P\left[\delta_v \leq \frac{c \log n}{\beta}\right] = 1 - \left(1 - e^{-c \log n}\right) = \frac{1}{n^c}$

Union bound over all vertices: $P\left[\delta_{\max} > \frac{c \log n}{\beta}\right] \leq \frac{1}{n^{c-1}}$

Parallel Low Diameter Decomposition [MPX SPAA ‘13]

- **Analysis of Number of Edges between Clusters:**

Parallel Low Diameter Decomposition [MPX SPAA ‘13]

- **Analysis of Number of Edges between Clusters:**
  - βm edges cut **in expectation**

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Analysis of Number of Edges between Clusters:**
  - βm edges cut **in expectation**
  - Show that probability for any edge (u, v), when fixing the midpoint w, the probability of the smallest and second smallest value of $\{T_v + d(v, w) \mid v \in V\}$ differs by at most 1 is upper bounded by β

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Analysis of Number of Edges between Clusters:**
  - βm edges cut **in expectation**
  - Show that probability for any edge (u, v), when fixing the midpoint w, the probability of the smallest and second smallest value of $\{T_v + d(v, w) \mid v \in V\}$ differs by at most 1 is upper bounded by β
    - Why is this enough?

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Analysis of Number of Edges between Clusters:**
  - βm edges cut **in expectation**
  - Show that probability for any edge (u, v), when fixing the midpoint w, the probability of the smallest and second smallest value of $\{T_v + d(v, w) \mid v \in V\}$ differs by at most 1 is upper bounded by β
    - Why is this enough?
    - **Only time edge is between clusters**

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Analysis of Number of Edges between Clusters:**
  - βm edges cut **in expectation**
  - Show that probability for any edge (u, v), when fixing the midpoint w, the probability of the smallest and second smallest value of $\{T_v + d(v, w) \mid v \in V\}$ differs by at most 1 is upper bounded by β
    - Why is this enough?
    - **Only time edge is between clusters**
  - Due to memoryless property, can use CDF $c(x) = 1 - e^{-\beta x}$

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Analysis of Number of Edges between Clusters:**
  - βm edges cut **in expectation**
  - Show that probability for any edge (u, v), when fixing the midpoint w, the probability of the smallest and second smallest value of $\{T_v + d(v, w) \mid v \in V\}$ differs by at most 1 is upper bounded by β
    - Why is this enough?
    - **Only time edge is between clusters**
  - Due to memoryless property, can use CDF $c(x) = 1 - e^{-\beta x}$
    - Fix time for one endpoint, probability other endpoint within one of that time is at most $1 - e^{-\beta}$

Parallel Low Diameter Decomposition [MPX SPAA '13]

- **Analysis of Number of Edges between Clusters:**
  - Due to memoryless property, given CDF $c(x) = 1 - e^{-\beta x}$
    - Fix time for one endpoint, probability other endpoint within one of that time is at most $1 - e^{-\beta}$
    - $1 - e^{-\beta} < \beta$ for $\beta > 0$ using Taylor series
    - Hence, for **any edge marginal probability edge is in cut is $\boldsymbol{\beta}$**
    - Expected number of edges in cut: βm

Algorithms Engineering: Theory-in-Practice

- What is algorithm engineering?

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- What is algorithm engineering?
  - Algorithm design

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- What is algorithm engineering?
  - Algorithm design
  - Algorithm analysis

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- What is algorithm engineering?
  - Algorithm design
  - Algorithm analysis
  - **Algorithm implementation**

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- What is algorithm engineering?
  - Algorithm design
  - Algorithm analysis
  - **Algorithm implementation**
  - **Optimization**

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- What is algorithm engineering?
  - Algorithm design
  - Algorithm analysis
  - **Algorithm implementation**
  - **Optimization**
  - **Profiling**

Uses slides from MIT 6.506 Algorithms Engineering

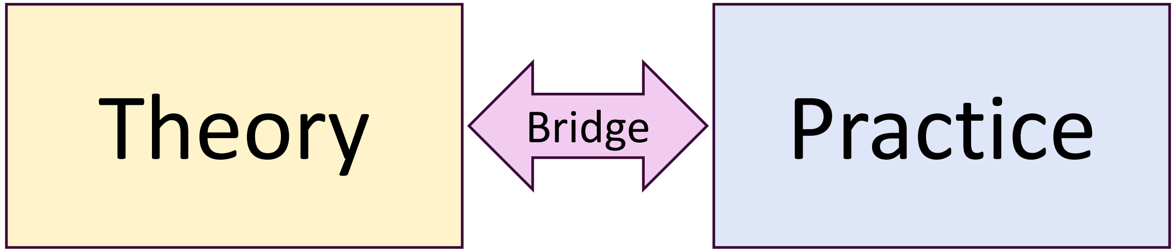
Algorithms Engineering: Theory-in-Practice

- What is algorithm engineering?
  - Algorithm design
  - Algorithm analysis
  - **Algorithm implementation**
  - **Optimization**
  - **Profiling**
  - **Experimental evaluation**

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

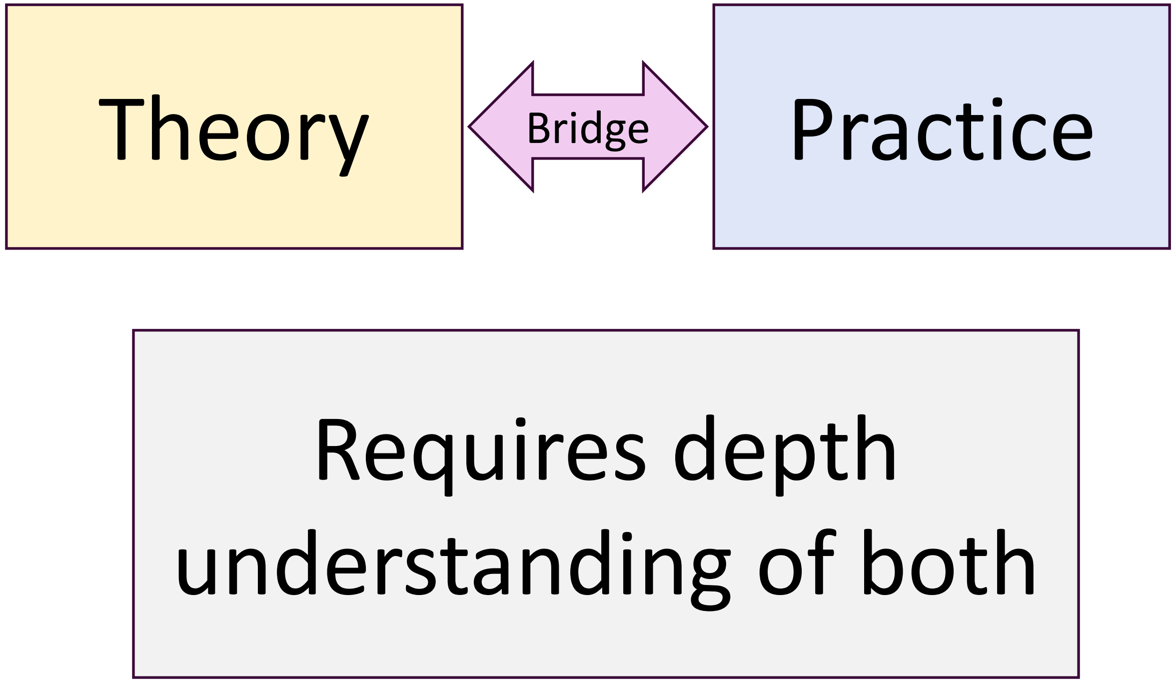
- What is algorithm engineering?
  - Algorithm design
  - Algorithm analysis
  - **Algorithm implementation**
  - **Optimization**
  - **Profiling**
  - **Experimental evaluation**



Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- What is algorithm engineering?
  - Algorithm design
  - Algorithm analysis
  - **Algorithm implementation**
  - **Optimization**
  - **Profiling**
  - **Experimental evaluation**



Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- How does one bridge theory and practice?

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- How does one bridge theory and practice?
  - **Use empirical findings to inform theory**

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- How does one bridge theory and practice?
  - **Use empirical findings to inform theory**
  - **Use theory to build confidence that algorithms will perform well in many different settings**
    - Heuristics do not guarantee this!

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- How does one bridge theory and practice?
  - **Use empirical findings to inform theory**
  - **Use theory to build confidence that algorithms will perform well in many different settings**
    - Heuristics do not guarantee this!
  - Ability to **predict performance** (e.g. in real-time applications)

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- How does one bridge theory and practice?
  - **Use empirical findings to inform theory**
  - **Use theory to build confidence that algorithms will perform well in many different settings**
    - Heuristics do not guarantee this!
  - Ability to **predict performance** (e.g. in real-time applications)
  - Develop **good theoretical models** that model real architectures

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- A history:
  - Early days, standard practice to implement algorithms you design

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- A history:
  - Early days, standard practice to implement algorithms you design
  - 1970s-1980s: Algorithm theory developed as a sub-discipline in CS used for “paper and pencil” work

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- A history:
  - Early days, standard practice to implement algorithms you design
  - 1970s-1980s: Algorithm theory developed as a sub-discipline in CS used for “paper and pencil” work
  - Late 1980s-1990s: Researchers began noticing gaps between theory and practice

Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- A history:
  - Early days, standard practice to implement algorithms you design
  - 1970s-1980s: Algorithm theory developed as a sub-discipline in CS used for "paper and pencil" work
  - Late 1980s-1990s: Researchers began noticing gaps between theory and practice
  - 1997: First Workshop on Algorithm Engineering by P. Italiano (now part of ESA)

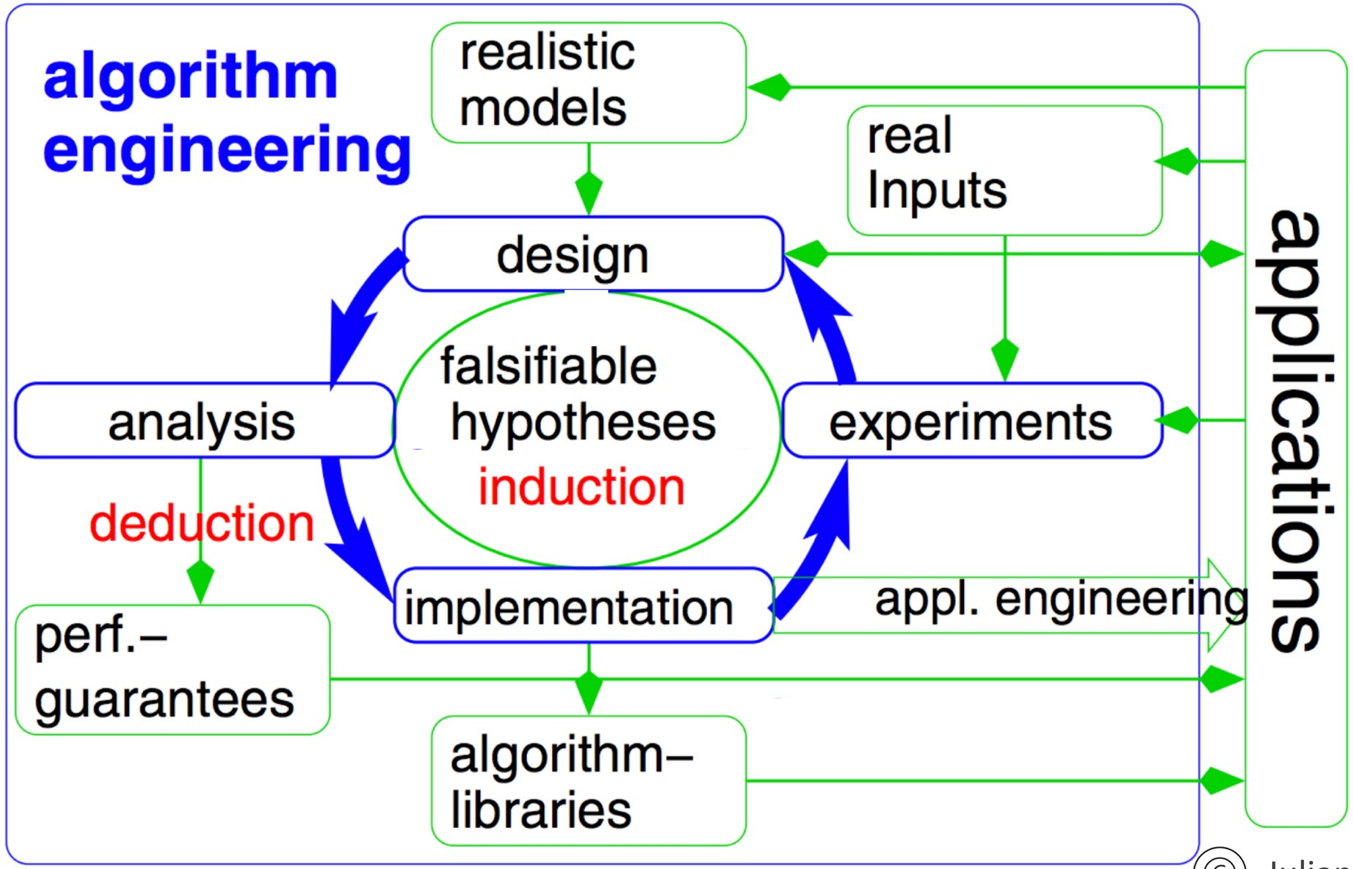
Uses slides from MIT 6.506 Algorithms Engineering

Algorithms Engineering: Theory-in-Practice

- A history:
  - Early days, standard practice to implement algorithms you design
  - 1970s-1980s: Algorithm theory developed as a sub-discipline in CS used for “paper and pencil” work
  - Late 1980s-1990s: Researchers began noticing gaps between theory and practice
  - 1997: First Workshop on Algorithm Engineering by P. Italiano (now part of ESA)
  - Now, many conferences on algorithms engineering ALENEX, SEA, ICML, NeurIPS, IJCAI etc.

Uses slides from MIT 6.506 Algorithms Engineering

What is Algorithm Engineering?



© Julian Shun

Uses slides from MIT 6.506 Algorithms Engineering

Source: "Algorithm Engineering – An Attempt at a Definition", Peter Sanders

Models of Computation

- Random-Access Machine (RAM)
  - Infinite Memory

Uses slides from MIT 6.506 Algorithms Engineering

Models of Computation

- Random-Access Machine (RAM)
  - Infinite Memory
  - Arithmetic operations, logical operations, and memory accesses take $O(1)$ time

Uses slides from MIT 6.506 Algorithms Engineering

Models of Computation

- Random-Access Machine (RAM)
  - Infinite Memory
  - Arithmetic operations, logical operations, and memory accesses take $O(1)$ time
  - Most sequential algorithms are designed in this model (intro to algorithms courses)

Uses slides from MIT 6.506 Algorithms Engineering

Models of Computation

- Random-Access Machine (RAM)
  - Infinite Memory
  - Arithmetic operations, logical operations, and memory accesses take $O(1)$ time
  - Most sequential algorithms are designed in this model (intro to algorithms courses)
- **Nowadays computers are much more complex**

Uses slides from MIT 6.506 Algorithms Engineering

Models of Computation

- **Nowadays computers are much more complex**
  - Deep cache hierarchies

Uses slides from MIT 6.506 Algorithms Engineering

Models of Computation

- **Nowadays computers are much more complex**
  - Deep cache hierarchies
  - Instruction level parallelism

Uses slides from MIT 6.506 Algorithms Engineering

Models of Computation

- **Nowadays computers are much more complex**
  - Deep cache hierarchies
  - Instruction level parallelism
  - Multiple cores

Uses slides from MIT 6.506 Algorithms Engineering

Models of Computation

- **Nowadays computers are much more complex**
  - Deep cache hierarchies
  - Instruction level parallelism
  - Multiple cores
  - Multiple machines

Uses slides from MIT 6.506 Algorithms Engineering

Models of Computation

- **Nowadays computers are much more complex**
  - Deep cache hierarchies
  - Instruction level parallelism
  - Multiple cores
  - Multiple machines
  - Disk if input doesn't fit in main memory

Uses slides from MIT 6.506 Algorithms Engineering

Models of Computation

- **Nowadays computers are much more complex**
  - Deep cache hierarchies
  - Instruction level parallelism
  - Multiple cores
  - Multiple machines
  - Disk if input doesn't fit in main memory
  - Asymmetric read-write costs in non-volatile memory

Uses slides from MIT 6.506 Algorithms Engineering

Algorithm Design and Analysis

- **Difference in asymptotic complexities:**
  - **Algorithm 1:** $N \log_2(N)$
  - **Algorithm 2:** $1000\,N$
  - **Which one is practically better?**

Uses slides from MIT 6.506 Algorithms Engineering

Algorithm Design and Analysis

- **Difference in asymptotic complexities:**
  - **Algorithm 1:** $\boldsymbol{N \log_2(N)}$
  - **Algorithm 2:** $\mathbf{1000}\ \boldsymbol{N}$

- **First, algorithm,** $\log_2(\#\ \text{of particles in the universe}) < \mathbf{300}$

Uses slides from MIT 6.506 Algorithms Engineering

Algorithm Design and Analysis

- **Difference in asymptotic complexities:**
  - **Algorithm 1:** $\boldsymbol{N \log_2(N)}$
  - **Algorithm 2:** $\mathbf{1000}\,\boldsymbol{N}$

- **First, algorithm,** $\log_2(\#\text{ of particles in the universe}) < \mathbf{300}$
- **Constant factors matter!**

Uses slides from MIT 6.506 Algorithms Engineering

Algorithm Design and Analysis

- **Difference in asymptotic complexities:**
  - **Algorithm 1:** $\boldsymbol{N \log_2(N)}$
  - **Algorithm 2:** $\boldsymbol{1000\, N}$

  - **First, algorithm,** $\log_2(\#\text{ of particles in the universe}) < \mathbf{300}$
  - **Constant factors matter!**
  - **Avoid unnecessary computations**

Uses slides from MIT 6.506 Algorithms Engineering

Algorithm Design and Analysis

- **Difference in asymptotic complexities:**
  - **Algorithm 1:** $\boldsymbol{N \log_2(N)}$
  - **Algorithm 2:** $\mathbf{1000}\ \boldsymbol{N}$

  - **First, algorithm,** $\log_2(\#\text{ of particles in the universe}) < \mathbf{300}$
  - **Constant factors matter!**
  - **Avoid unnecessary computations**
  - **Simplicity** improves practicality and leads to better performance

Uses slides from MIT 6.506 Algorithms Engineering

Algorithm Design and Analysis

- **Consider tradeoffs:**
  - Time vs. space tradeoffs
  - Work vs. parallelism tradeoffs

Uses slides from MIT 6.506 Algorithms Engineering

Implementations

- **Write clean, modular code**
- **Write correctness checkers**
- **Save previous versions of your code!**

Uses slides from MIT 6.506 Algorithms Engineering

Implementations

- **Write clean, modular code**
  - Easier to experiment with different methods, and save a lot of development time when adding new features
- **Write correctness checkers**
  - Especially important in numerical and geometric applications because of floating-point arithmetic errors and non-determinism, different results from different runs!
- **Save previous versions of your code!**
  - Version control always! If you need to rollback changes!

Uses slides from MIT 6.506 Algorithms Engineering

Experimentation

- Test code using **timers** and **performance profilers**
  - Perf, gprof, valgrind

Uses slides from MIT 6.506 Algorithms Engineering

Experimentation

- Test code using **timers** and **performance profilers**
  - Perf, gprof, valgrind
- Use **large variety of inputs** (both real-world and synthetic)—ensures implementation is robust for many different environments

Uses slides from MIT 6.506 Algorithms Engineering

Experimentation

- Test code using **timers** and **performance profilers**
  - Perf, gprof, valgrind
- Use **large variety of inputs** (both real-world and synthetic)—ensures implementation is robust for many different environments
  - Use different sizes

Uses slides from MIT 6.506 Algorithms Engineering

Experimentation

- Test code using **timers** and **performance profilers**
  - Perf, gprof, valgrind
- Use **large variety of inputs** (both real-world and synthetic)—ensures implementation is robust for many different environments
  - Use different sizes
  - Use worst-case inputs to identify correctness or performance issues

Experimentation

- Test code using **timers** and **performance profilers**
  - Perf, gprof, valgrind
- Use **large variety of inputs** (both real-world and synthetic)—ensures implementation is robust for many different environments
  - Use different sizes
  - Use worst-case inputs to identify correctness or performance issues
- **Reproducibility**

Uses slides from MIT 6.506 Algorithms Engineering

Experimentation

- **Reproducibility**
  - Document environmental setup

Uses slides from MIT 6.506 Algorithms Engineering

Experimentation

- **Reproducibility**
  - Document environmental setup
  - Fix random seeds if needed

Experimentation

- **Reproducibility**
  - Document environmental setup
  - Fix random seeds if needed
- Run **multiple timings** to determine with variance

Uses slides from MIT 6.506 Algorithms Engineering

Experimentation

- **Reproducibility**
  - Document environmental setup
  - Fix random seeds if needed
- Run **multiple timings** to determine with variance
- For parallel code, test on varying numbers of processors to study scalability

Experimentation

- **Reproducibility**
  - Document environmental setup
  - Fix random seeds if needed
- Run **multiple timings** to determine with variance
- For parallel code, test on varying numbers of processors to study scalability
- Compare with best sequential code

Uses slides from MIT 6.506 Algorithms Engineering

Experimentation

- **Reproducibility**
  - Document environmental setup
  - Fix random seeds if needed
- Run **multiple timings** to determine with variance
- For parallel code, test on varying numbers of processors to study scalability
- Compare with best sequential code
- For reproducibility, write **deterministic parallel code** if possible

Uses slides from MIT 6.506 Algorithms Engineering

Experimentation

- **Reproducibility**
  - Document environmental setup
  - Fix random seeds if needed
- Run **multiple timings** to determine with variance
- For parallel code, test on varying numbers of processors to study scalability
- Compare with best sequential code
- For reproducibility, write **deterministic parallel code** if possible
- **Useful benchmarking tools:** Cilkscale, Cilksan

Uses slides from MIT 6.506 Algorithms Engineering

Libraries and Frameworks are Very Important!

- **Consistency in experimental environment**
- **Easier development in the future—no wasted work in development**

Uses slides from MIT 6.506 Algorithms Engineering

Libraries and Frameworks are Very Important!

- **Consistency in experimental environment**
- **Easier development in the future—no wasted work in development**
- **Service to the community to develop your own for your needs and potential other needs in the future**

Uses slides from MIT 6.506 Algorithms Engineering

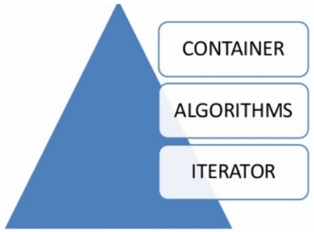
Libraries and Frameworks are Very Important!

- **Example frameworks:**

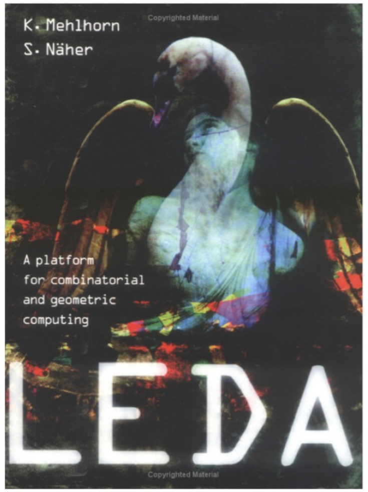
ParlayLib - A Toolkit for Programming Parallel Algorithms on Shared-Memory Multicore Machines

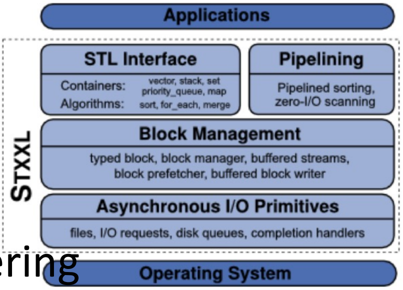
Build passing | codecov 97% | License MIT

© Julian Shun



CGAL





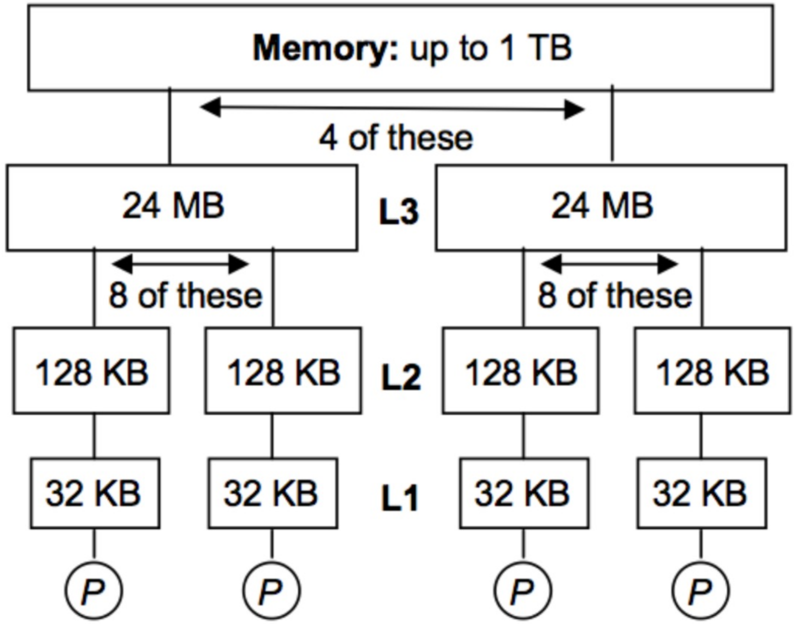
boost C++ LIBRARIES

Problem Based Benchmark Suite

Home | Benchmarks/Code | Inputs | License | People | Publications

Uses slides from MIT 6.506 Algorithms Engineering

Cache Hierarchies



Design cache-efficient and cache-oblivious algorithms to improve locality

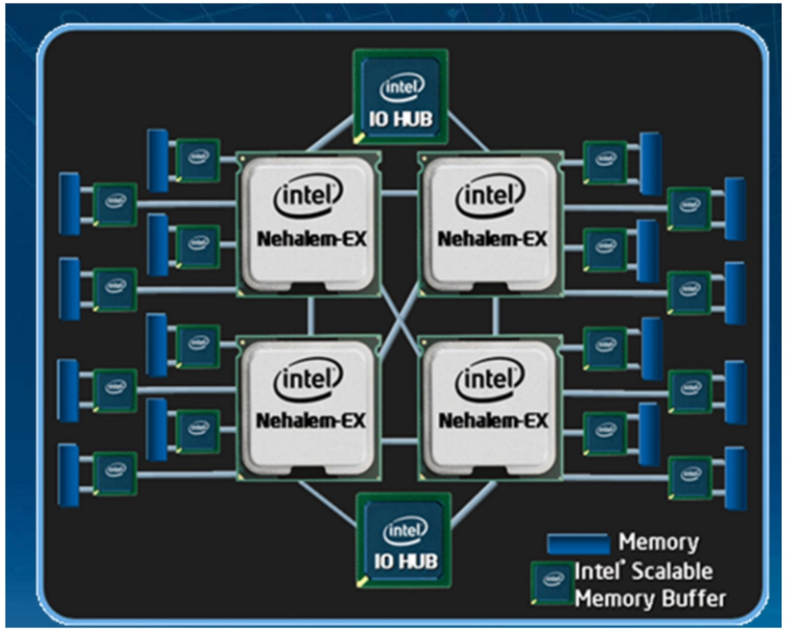
© Julian Shun

Memory level	Approx latency
L1 Cache	1-2ns
L2 Cache	3-5ns
L3 cache	12-40ns
DRAM	60-100ns

Uses slides from MIT 6.506 Algorithms Engineering

Non-Uniform Memory Access (NUMA)

- Accessing **remote memory** is more expensive than accessing local memory of a socket
  - Latency depends on the number of hops



Design NUMA-aware algorithms to improve locality

© Julian Shun

Uses slides from MIT 6.506 Algorithms Engineering

I/O Efficiency

- Need to read input from disk at least once
- Need to read many more times if input doesn't fit in memory

Memory	Latency	Throughput
DRAM	60–100 ns	Tens of GB/s
SSD	Tens of μs	500 MB–2 GB/s (seq), 50–200 MB/s (rand)
HDD	Tens of ms	200 MB/s (seq), 1 MB/s (rand)

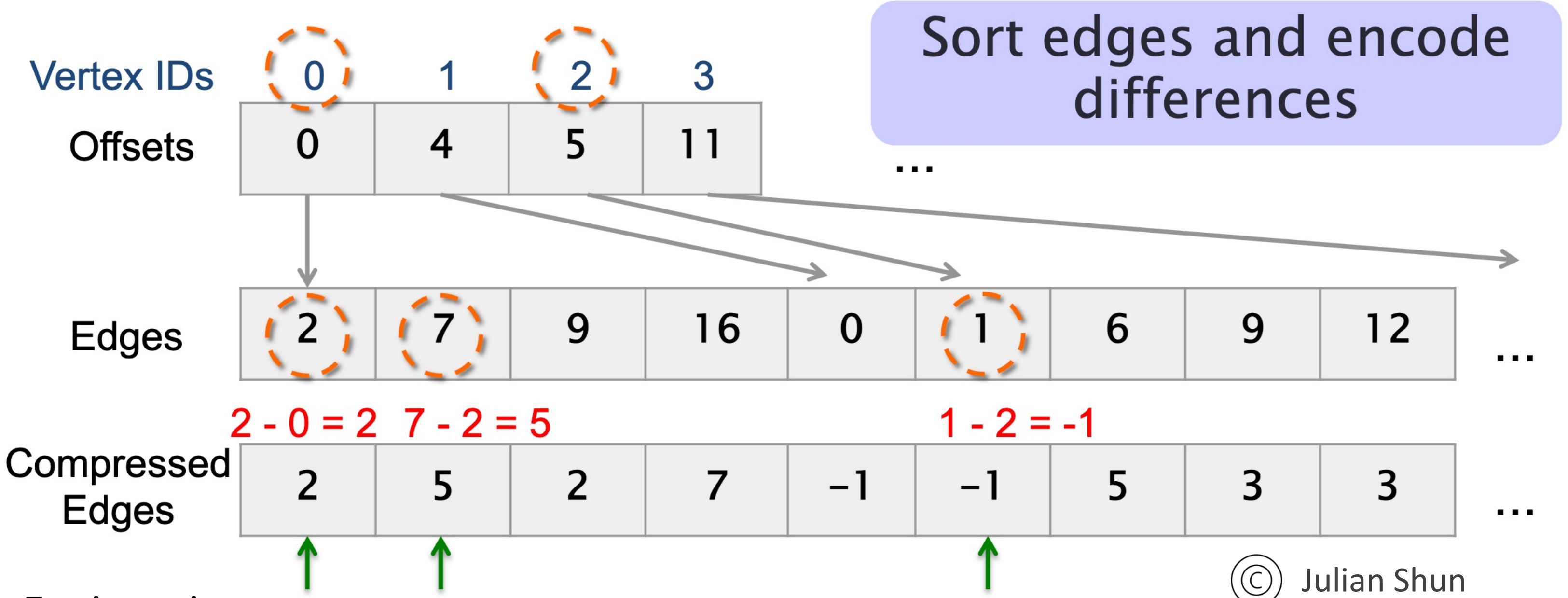
Uses slides from MIT 6.506 Algorithms Engineering

Very Large Graphs!

- Need **graph compression** if graph cannot fit on machine

Very Large Graphs!

- Need **graph compression** if graph cannot fit on machine
- **Compressed Sparse Row (CSR), Compressed Row Storage (CRS), Yale format**



Uses slides from MIT 6.506 Algorithms Engineering

Many Things in Theory and Practice

- Many things to learn in theory and practice
- **Lots of knowledge in between**
- **Communication between the communities important**
  - **Will lead to practical impact!**