A General-purpose Distributed and Parallel Programming System at Scale

Dr. Tsung-Wei Huang

Department of Electrical and Computer Engineering

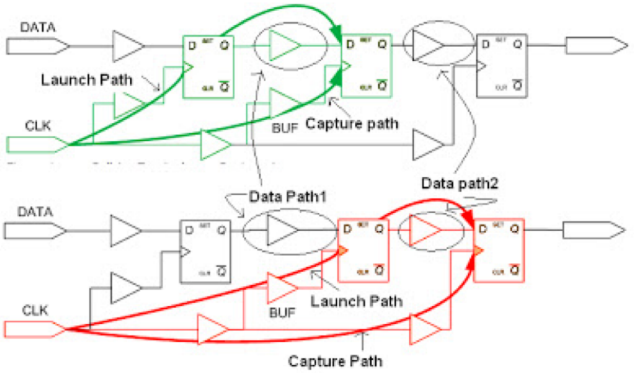
University of Illinois at Urbana-Champaign, IL, USA

Outline

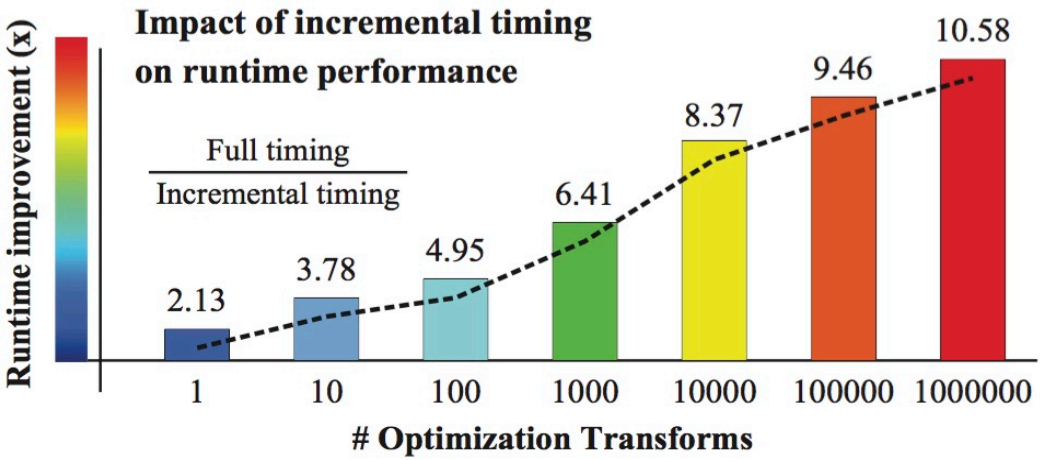
- **My early PhD research**
  - OpenTimer: A VLSI timing analysis tool
  - Express parallelism in the right way
- **A general-purpose distributed programming system**
  - DtCraft and its ecosystem
  - Machine learning, graph algorithms, chip design
  - Cpp-Taskflow: Modern C++ parallel task programming
- **Conclusion and future work**

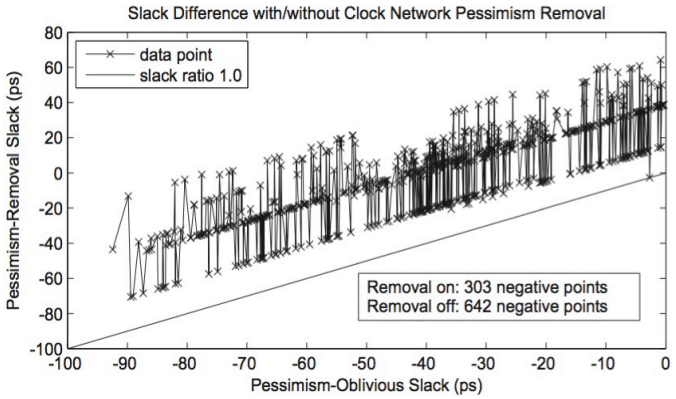
My Early PhD Research

- **Static timing analysis (STA)**
  - Key component in VLSI design
  - Verify the circuit timing
- **Many challenges**
  - Incremental timing
  - Parallelization, scalability



Setup/hold timing checks

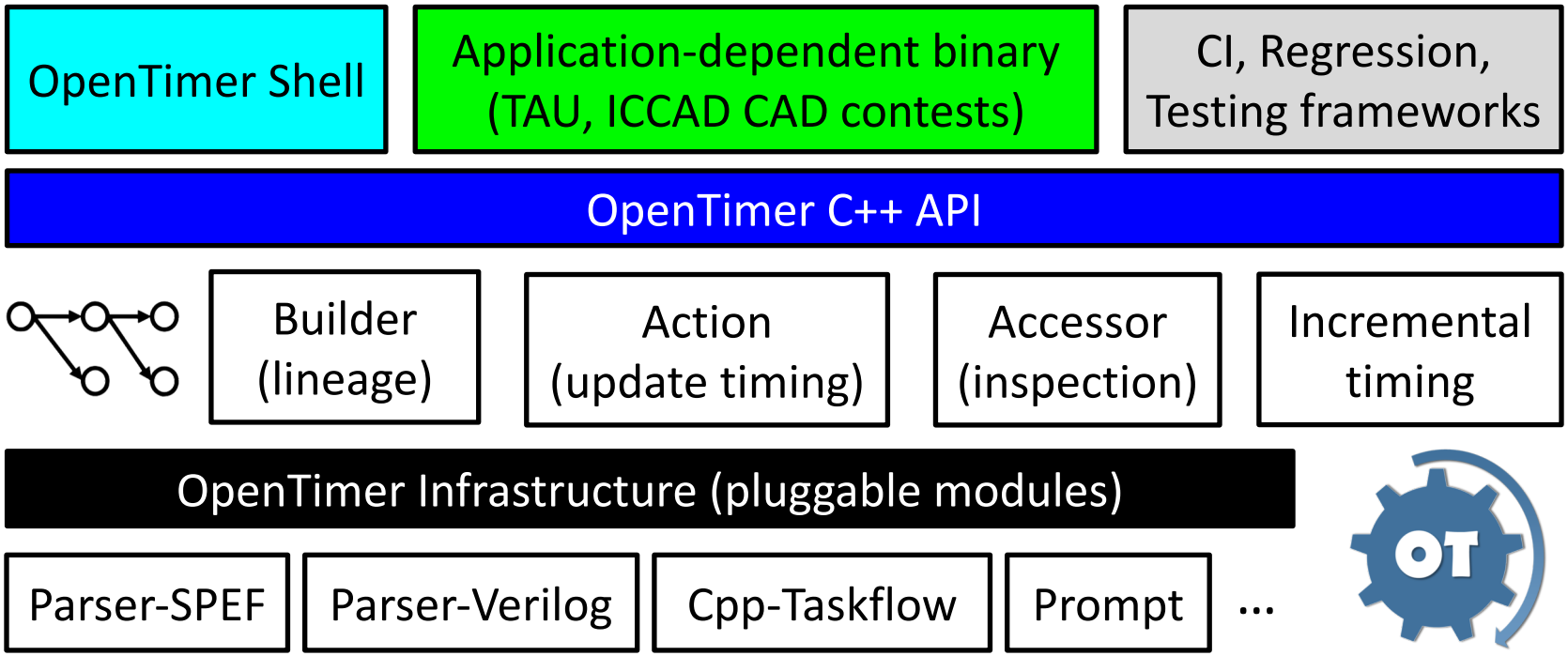




Path-based timing analysis to reduce pessimism

OpenTimer: An Open-source STA Tool

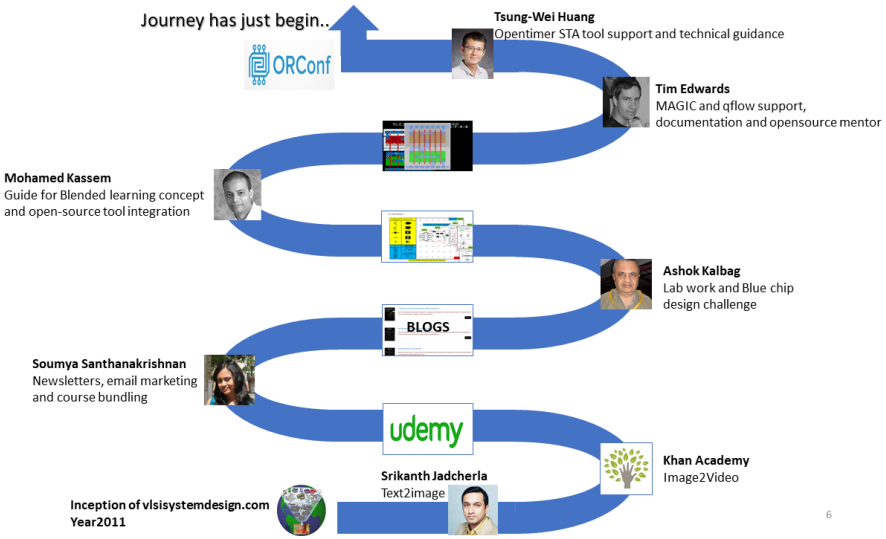
- TAU14 (1st place), TAU15 (2nd place), TAU16 (1st place)
- Best tool awards (WOSET ICCAD18, LibreCore16)
- Golden timers in VLSI and CAD communities



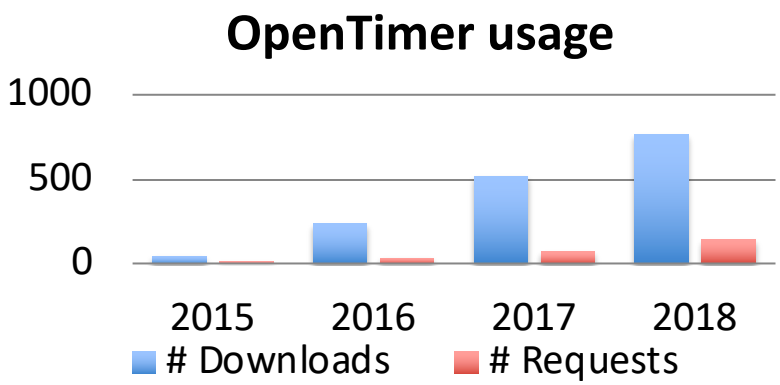
T.-W. Huang et al., "OpenTimer: A High-performance Timing Analysis Tool," IEEE/ACM ICCAD15

OpenTimer Community – Thank You!

OpenTimer is begin used in many projects



Collaboration with VSD (start-up) and Udemy online education platform



Song Chen *Contest Committee Member*

Xin Zhao *Contest Committee Member*

Xi Chen *Contest Committee Member*

Debjit Sinha *Workshop General Chair*

Qiuyang Wu *Workshop Technical Chair*

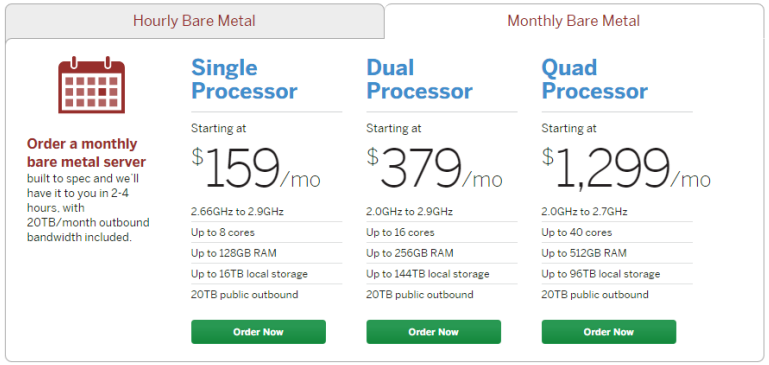
Tsung-Wei Huang *OpenTimer Support*

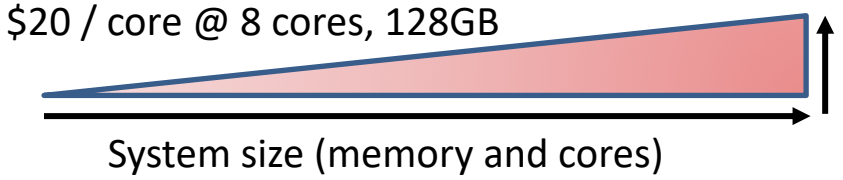
Acknowledged by ACM TAU16-19 Contests

We are growing! ***> 500 downloads in 2018***

Collaboration with IBM

- ***"We need a new distributed timing analysis method to deal with the large design complexity,"* IBM Timing Group, Fishkill, NY, 2015**
  - Existing tools are architecturally limited by one machine
  - Can take *800GB RAM* and *14 days* on a single machine
  - Maintaining "bare-metal" machines is not cost-efficient





Objective: Execute the same # computations, distributed across an array of less expensive per unit cost cloud systems

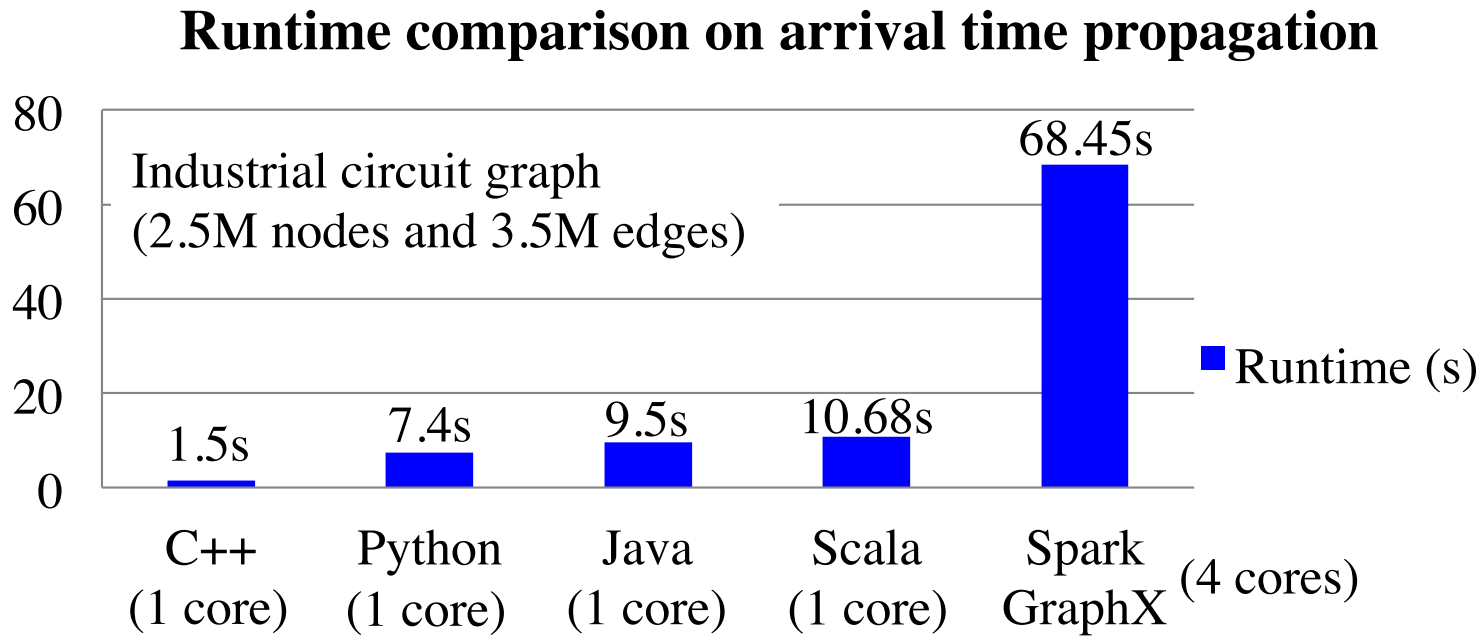
Leverage economy of scale by decreasing the per unit cost of computation "hill"

Existing Tools in Cloud Computing

- **Hadoop**
  - Distributed MapReduce platform on HDFS
- **Zookeeper**
  - Coordination service for distributed application
- **Mesos**
  - A high-performance cluster manager
- **Spark**
  - A general computing engine for big-data analytics
- **…**

Are these tools suitable for our application?

Big-data Tools is NOT an Easy Fit …

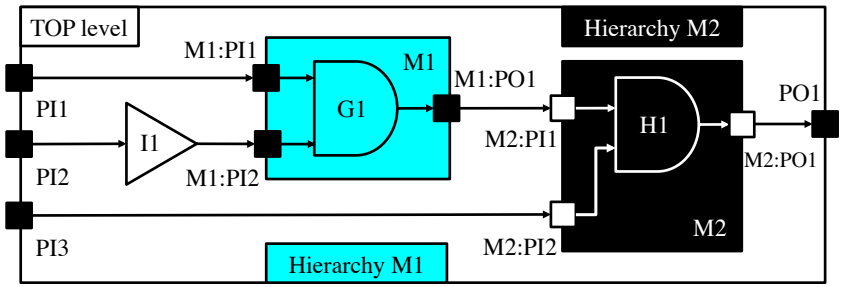


Method	Spark 2.0 (RDD + GraphX Pregel)	Java (SP)	C++ (SP)
Runtime (s)	68.45	9.5	**1.50**

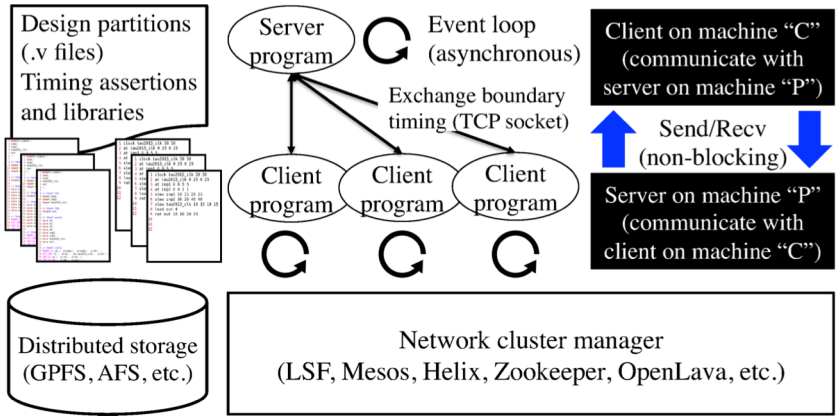
MapReduce overhead *Language overhead*

If it is Hard, let me Hard-code it …

- **A "specialized" distributed timer**
  - Low-level Linux sockets and message passing code
  - Explicit resource partition and mapping
- **Single-server multiple-client model**
  - Server coordinates with clients to exchange data



Three partitions, top-level, M1, and M2 (given by design teams)



T.-W. Huang, et al, "A Distributed Timing Analysis Framework for Large Designs," IEEE/ACM DAC16
T.-W. Huang, et al, US Patent US20170242945A1, assignee IBM, 2017
T.-W. Huang, et al, US Patent US9836572B2, assignee IBM, 2017

Observations

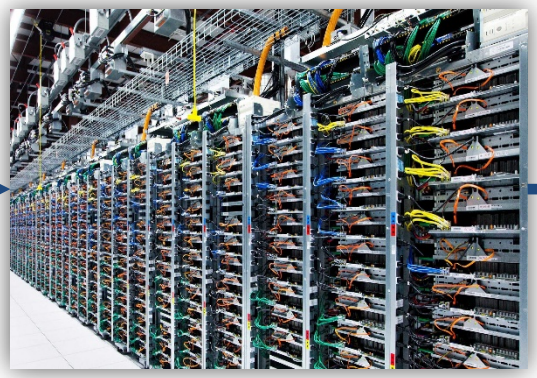
- Existing big data tools have large room to improve
- Fast prototype but hard long-term maintenance
- Non-unified tooling environment
- Cumbersome software dependencies
- Cannot afford hard-coded solution all the time
- Difficult low-level concurrency controls
- Hand-written message passing code
- …

We want a general-purpose programming system to streamline the building of parallel/distributed systems!

DARPA's Electronic Resurgence Initiative

- **$1.5B investment to reduce the design complexity**
  - People without chip design experience can design chips

~~Chip layout army~~



Massive cloud computing!

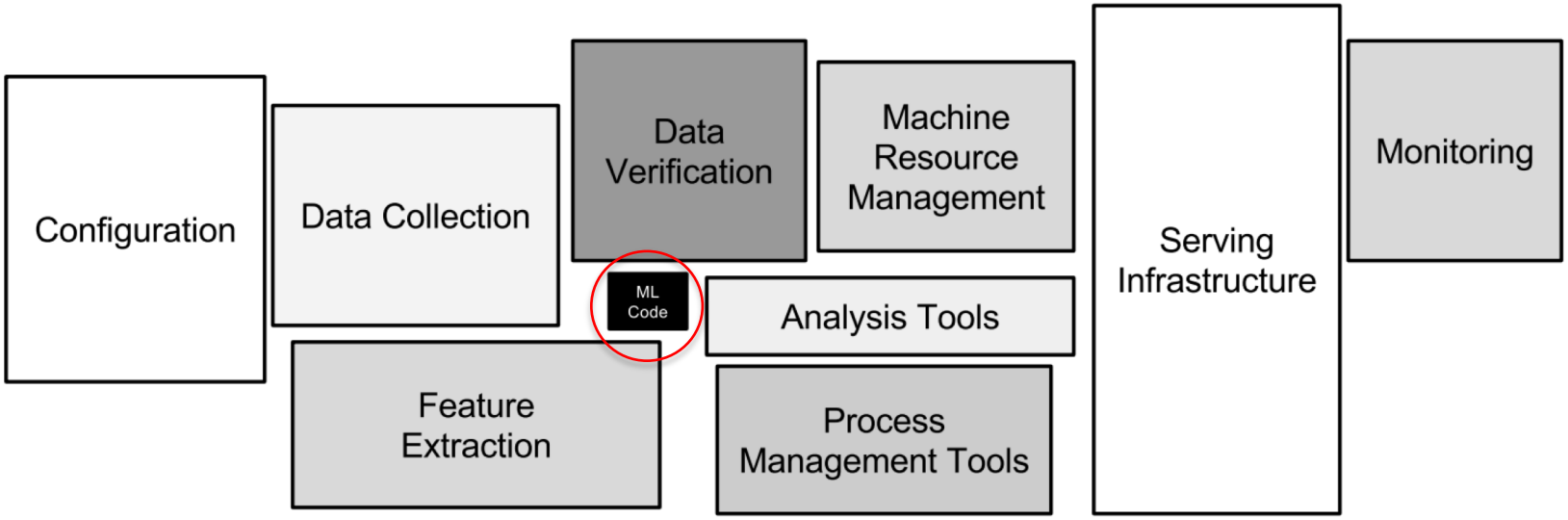
No-touch foundry

- **A general-purpose silicon compiler (IDEA & POSH)**
  - fully-automated layout generator, no human in the loop
  - Massive software efforts and tool integrations
  - *Parallel/distributed computing and machine learning**

**DARPA IDEA/POSH Kick-off Meeting, Boston, June, 2018*

Hidden Technical Debts in ML Systems

- **Only a TINY fractions of ML systems is ML code**
  - Writing ML code is cheap; system maintenance is $$$$$
- **ML innovations are moving to "system innovations"**
  - Spark MLflow, TensorFlow 2.0, PyTorch, MindsDB
  - *Parallel/distributed computing, transparency, scalability*



Google Inc, "Hidden Technical Debt in Machine Learning Systems," NIPS 2017

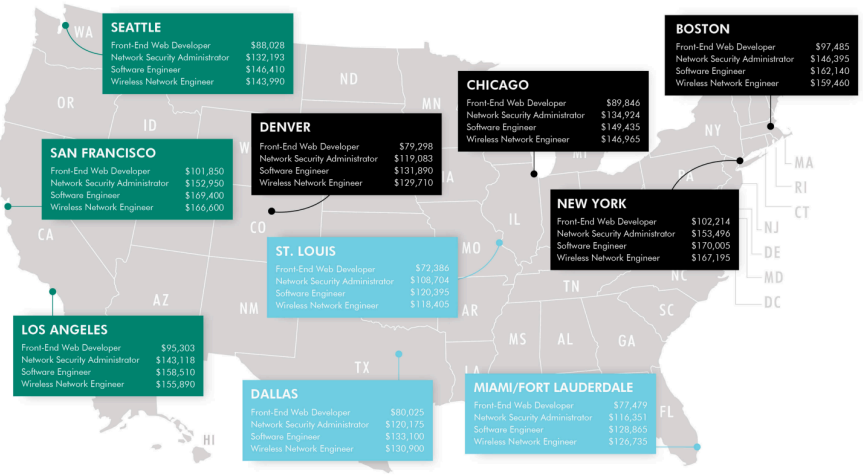
Keep Programmability in Mind

- **In the cloud era …**
  - Hardware is just a commodity
  - Building a cluster is cheap
  - Coding takes people and time

Google Cloud

TECH SALARIES ACROSS THE U.S.

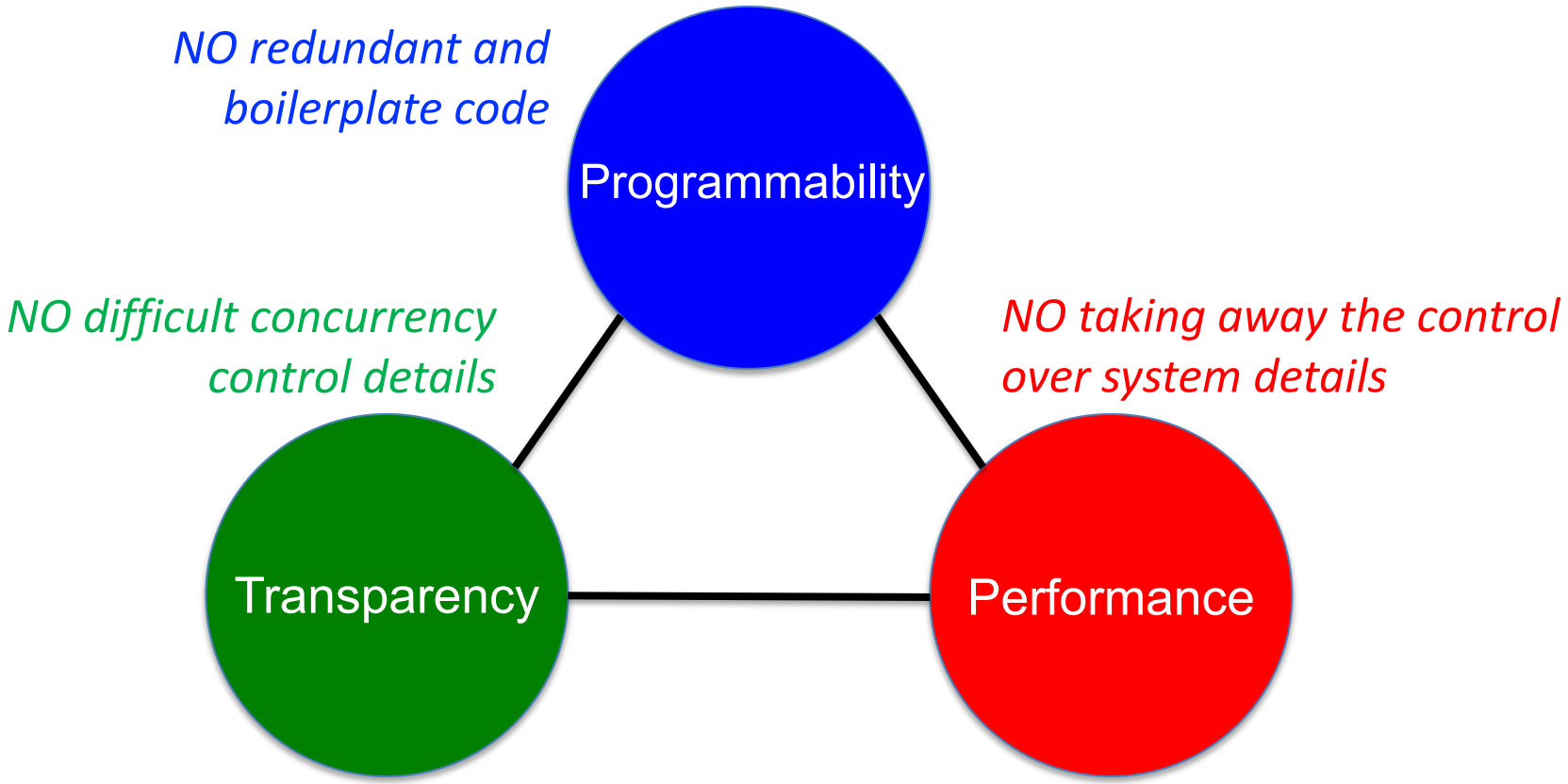
Compare midpoint starting salaries for technology positions in different cities.*



Programmability can affect the "performance" and "productivity" in all aspects (details, styles, high-level decisions, etc)!

2018 Avg Software Engineer salary (NY) > $170K

Want a Unified Programming System

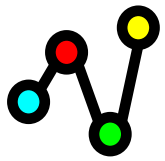


"We want to let users easily express their parallel/distributed computing workload without taking away the control over system details to achieve high performance, using our expressive API written in modern C++"

Outline

- **My early PhD research**
  - OpenTimer: A VLSI timing analysis tool
  - Express parallelism in the right way
- **A general-purpose distributed programming system**
  - DtCraft and its ecosystem
  - Machine learning, graph algorithms, chip design
  - Cpp-Taskflow: Modern C++ parallel task programming
- **Conclusion and future work**

A New Solution: DtCraft



Application pipeline (domain-specific applications)

MLCraft (distributed machine learning library)

GraphCraft (distributed graph processing library)

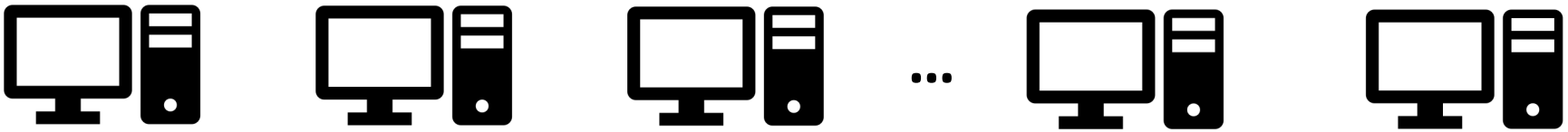
...

DataCraft (distributed data processing server)

DtCraft C++ Programming Runtime

Cpp-Taskflow (Parallel task programming library)

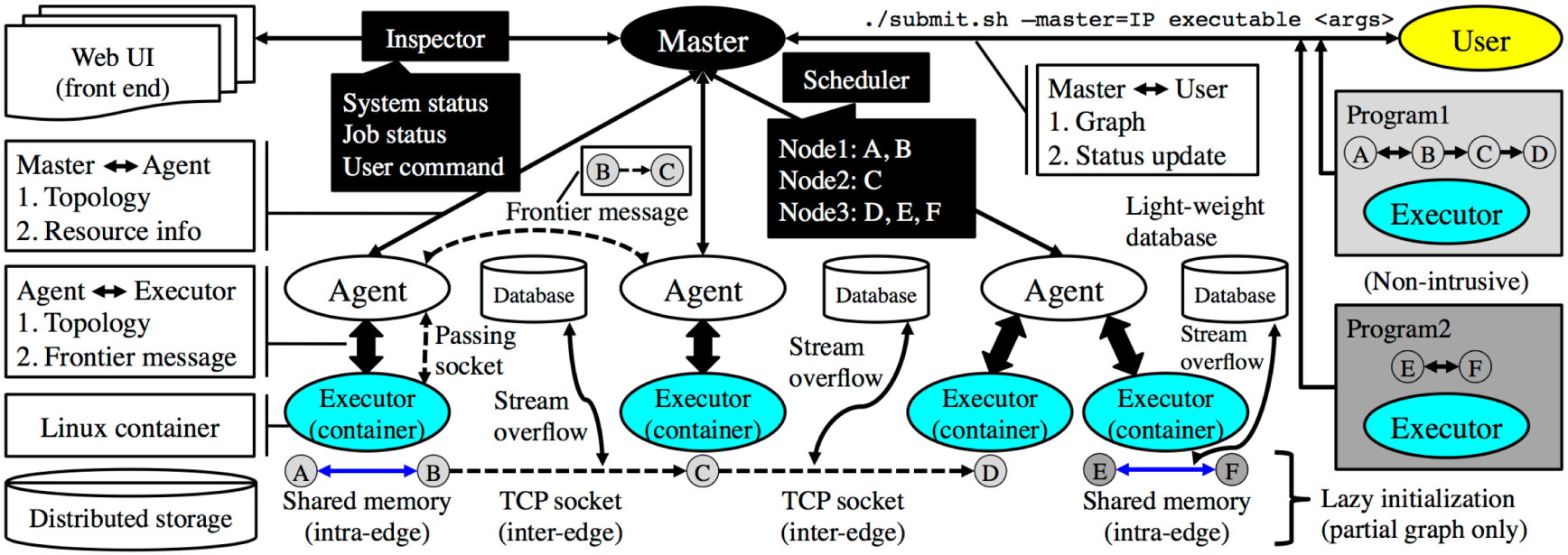
Cpp-Container (Docker, RunC, LibContainer, LXC)



DARPA IDEA Grant, NSF Grant, Best Open-source Tool Awards (IEEE IPDPS, ACM MM, WOSET, CPPConf, ACM DEBS, ACM/IEEE DAC, IEEE/ACM ICCAD, IEEE TCAD, ACM TAU, US Patents, etc.)

DtCraft C++ Programming Runtime

- **You describe an application in a *stream graph***
  - "No arm wrestling" with difficult concurrency details
- **Everything is by default distributed**
  - Same program runs on one and many machines



A Hello-World Example

- **An iterative and incremental flow**
  - Two vertices + two streams

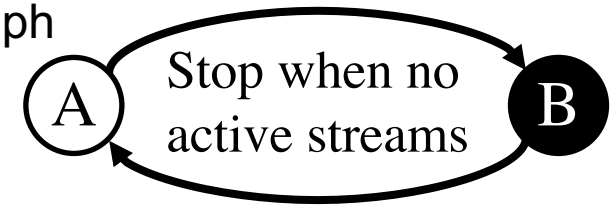
Step 4: A's resource
1 CPU / 1 GB RAM

Step 2:
string msg;

Step 3: A➔B callback

```
[=] (auto& B, auto& is) {
  Extract string from is;
  print string;
}
```

Step 1: stream graph



istream ➔ B

Step 3: A➔B callback

```
[=] (auto& A, auto& is) {
  Extract string from is;
  print string
}
```

Step 2:
string msg;

Step 4: B's resource
1 CPU / 1 GB RAM

A ⬅ istream

Step 5: ./submit –master=127.0.0.1 hello-world

DtCraft Code of Hello World

```
dtc::Graph G;

auto A = G.vertex();
auto B = G.vertex();

G.container().add(A).cpu(1).memory(1_GB);

auto AB = G.stream(A, B).on(
  [] (dtc::Vertex& B, dtc::InputStream& is) {
    if(std::string b; is(b) != -1) {
      dtc::cout("Received: ", b, '\n');
      return dtc::Event::REMOVE;
    }
    return dtc::Event::DEFAULT;
  }
);

auto BA = G.stream(B, A);

A.on([&AB] (dtc::Vertex& v) {
  (*v.ostream(AB))("hello world from A"s);
  dtc::cout("Sent 'hello world from A' to stream ", AB, "\n");
});

G.container().add(B).cpu(1).memory(1_GB);

B.on([&BA] (dtc::Vertex& v) {
  (*v.ostream(BA))("hello world from B"s);
  dtc::cout("Sent 'hello world from B' to stream ", BA, "\n");
});

BA.on([] (dtc::Vertex& A, dtc::InputStream& is) {
  if(std::string a; is(a) != -1) {
    dtc::cout("Received: ", a, "\n");
    return dtc::Event::REMOVE;
  }
  return dtc::Event::DEFAULT;
});

dtc::Executor(G).run();
```

- **Only a couple lines of code**
- Single sequential program
- No explicit data management
- Distributed across computers
- Easy-to-use interface
- Asynchronous by default
- Scalable to many threads
- Scalable to many machines
- In-context resource controls
- Transparent concurrency
- Automatic Linux container

… and more

Without DtCraft …

```cpp
auto count_A = 0;
auto count_B = 0;

// Send a random binary data to fd and add the
// received data to the counter.
auto pinpong(int fd, int& count) {
  auto data = random<bool>()
  auto w = write(fd, &data, sizeof(data));
  if(w == -1 && errno != EAGAIN) {
    throw system_error("Failed on write");
  }
  data = 0;
  auto r = read(fds, &data, sizeof(data));
  if(r == -1 && errno != EAGAIN) {
    throw system_error("Failed on read");
  }
  count += data;
}

int fd = -1;
std::error_code errc;
```

server.cpp

```cpp
if(getenv("MODE") == "SERVER") {
  fd = make_socket_server_fd("9999", errc);
}
else {
  fd = make_socket_client_fd("127.0.0.1", "9999", errc);
}
```

client.cpp

```cpp
if(fd == -1) {
  throw system_error("Failed to make socket");
}
```

Branch your code to server and client for distributed computation!

simple.cpp → server.cpp + client.cpp

```cpp
int make_socket_server_fd(
  std::string_view port,
  std::error_code errc
) {
  int fd {-1

  struct adc
  struct adc
  struct adc

  std::memse
  hints.ai_f
  hints.ai_s
  hints.ai_p
  hints.ai_f

  int one {1
  int ret;

  if((ret =
    errc = m
    return -
  }

  // Try to
  for(ptr =

    // Ignor
    if(ptr->
      goto t
    }

    if((fd =
      errc =
      goto t
    }

    ::setsoc

    if(::bir
      errc =
      goto t
    }

    if(::lis
      errc =
      goto t
    }
    else {
      break;
    }

  try_next
```

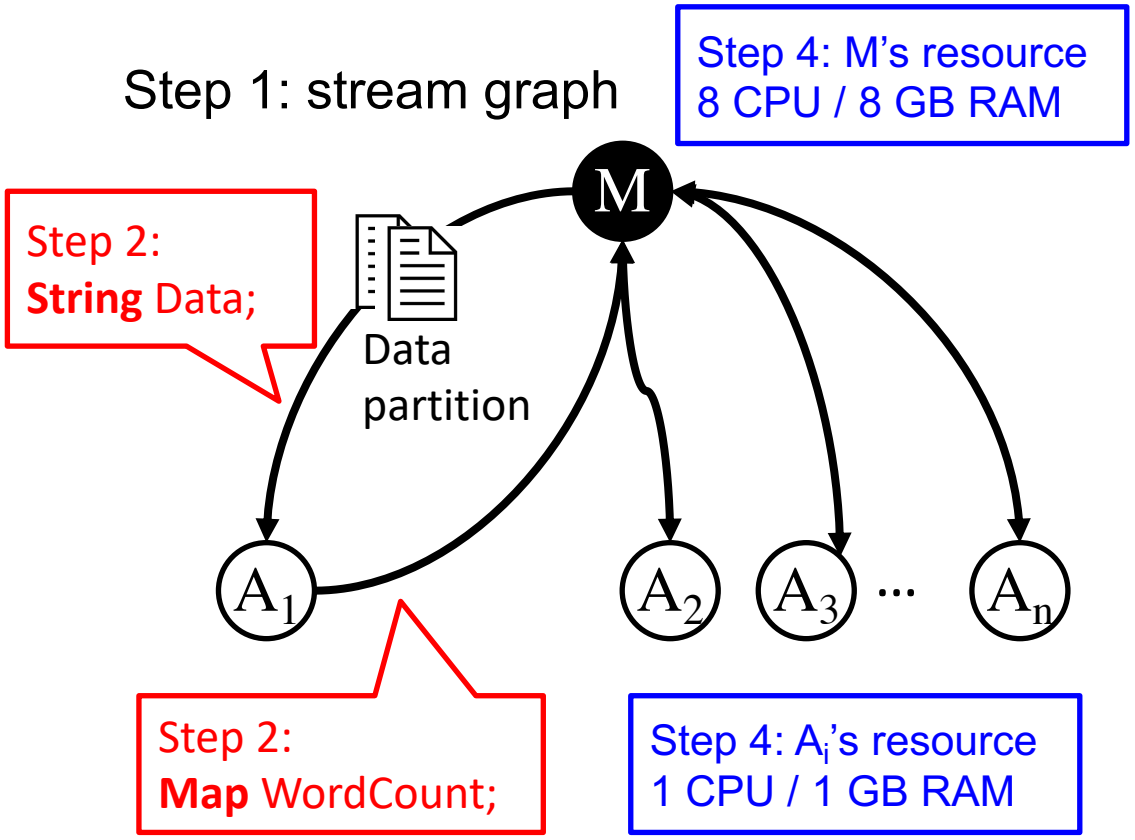
```cpp
    if(fd != -1) {
      ::close(fd);
      fd = -1;
    }
  }

  ::freeac

  // Assig
  return f
}

int make_s
  std::str
  std::str
  std::err
) noexcept

  errc.cle

  struct a
  struct a

  std::mem
  hints.ai
  hints.ai
  hints.ai

  int ret;
  int fd =
  int tri

  if((ret
    errc =
    return
  }

  // Try e
  for(auto

    // Ign
    if(ptr
      goto
    }

    if((fd
      errc
      goto
    }

  make_f
```

```cpp
    make_fd_close_on_exec(fd);

    tries = 3;

    issue_connect:
    ret = ::connect(fd, ptr->ai_addr, ptr->ai_addrlen);

    if(ret == -1) {
      if(errno == EINTR) {
        goto issue_connect;
      }
      else if(errno == EAGAIN && tries--) {
        std::this_thread::sleep_for(std::chrono::milliseconds(500));
        goto issue_connect;
      }
      else if(errno != EINPROGRESS) {
        goto try_next;
      }
      errc = make_posix_error_code(errno);
    }

    // Poll the socket. Note that writable return doesn't mean it is connec
    if(select_on_write(fd, 5, errc) && !errc) {
      int optval = -1;
      socklen_t optlen = sizeof(optval);
      if(::getsockopt(fd, SOL_SOCKET, SO_ERROR, &optval, &optlen) < 0) {
        errc = make_posix_error_code(errno);
        goto try_next;
      }
      if(optval != 0) {
        errc = make_posix_error_code(optval);
        goto try_next;
      }
      break;
    }

    try_next:

    if(fd != -1) {
      ::close(fd);
      fd = -1;
    }
  }

  ::freeaddrinfo(res);

  return fd;
}
```

A lot of boilerplate code plus hundred lines of scripts to enable distributed flow…

Word Counting MapReduce Example

- A master manages several workers



Step 5: ./submit –master=127.0.0.1 word-count

Step 3: A_i ← M callback

```
[=] (auto& Ai, auto& is) {
  Extract data from is;
  Count words in data;
  Store results in a map;
  Send the map to M;
}
```

A ⟸ istream

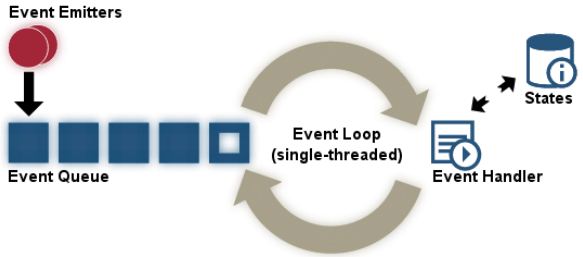
Step 3: A_i → M callback

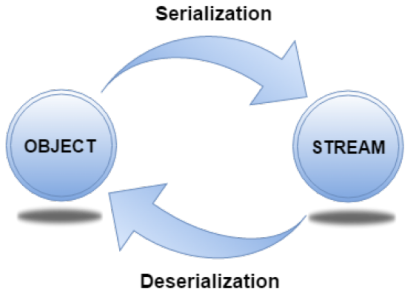
```
[=] (auto& M, auto& is) {
  Extract map from is;
  Reduce sum on map;
  if all done: print map;
  else: wait;
}
```

istream ⟹ M

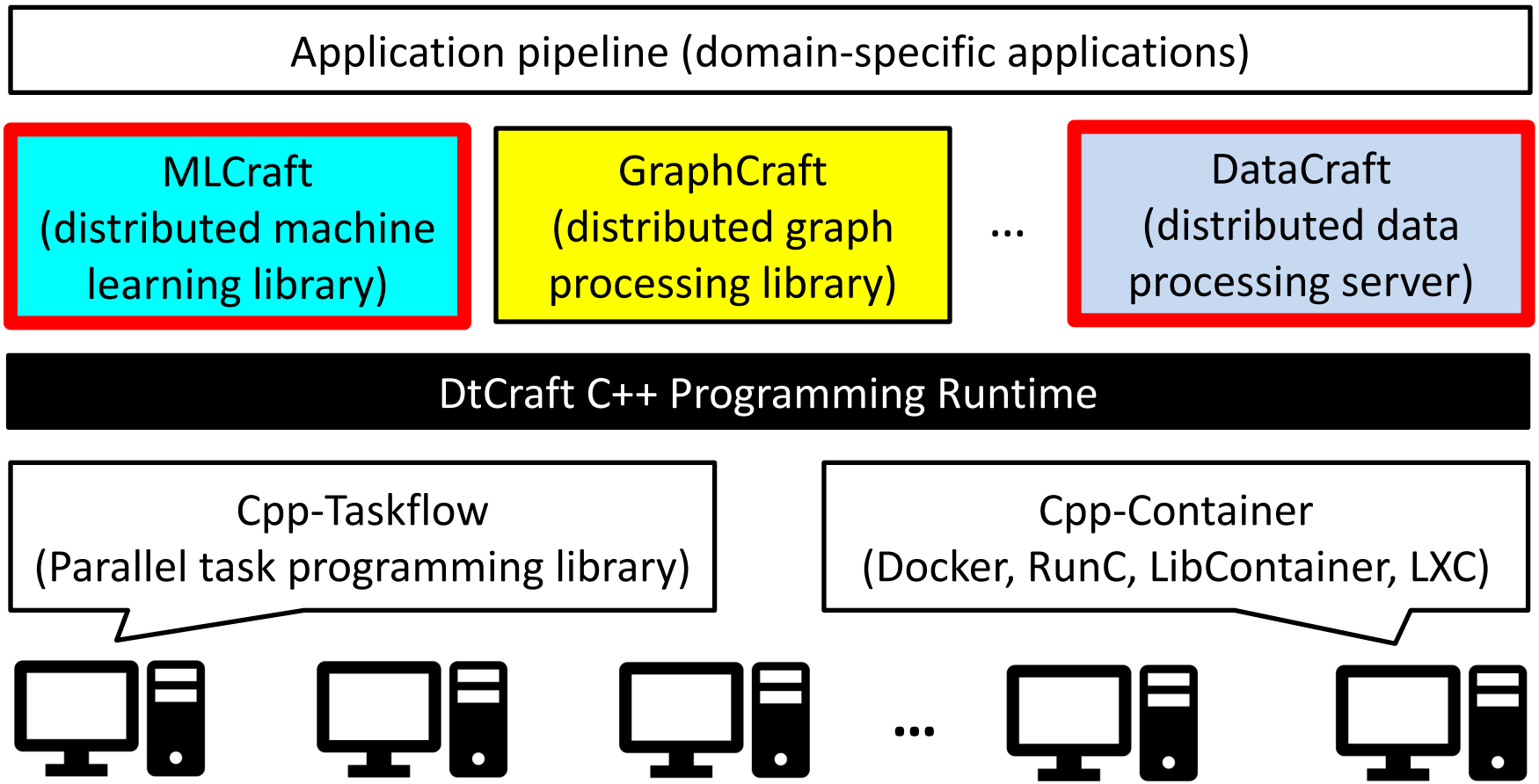
System Implementation Details

- **Kernel – Master, Agent, and Executor**
  - Master: global scheduler, deployment
  - Agent: local scheduler
  - Executor: task executor
- **Event-driven environment**
  - A new C++ reactor library
  - Thread-safe, lock-free, non-blocking IO
- **Streaming interface**
  - Built-in serialization library
- **Linux container**
  - Secure, safe, and robust resource control





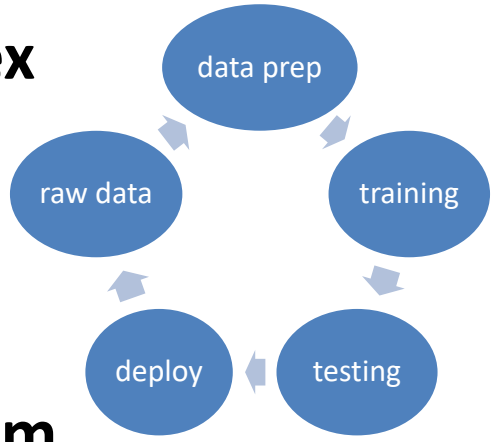
DtCraft to Handle Machine Learning

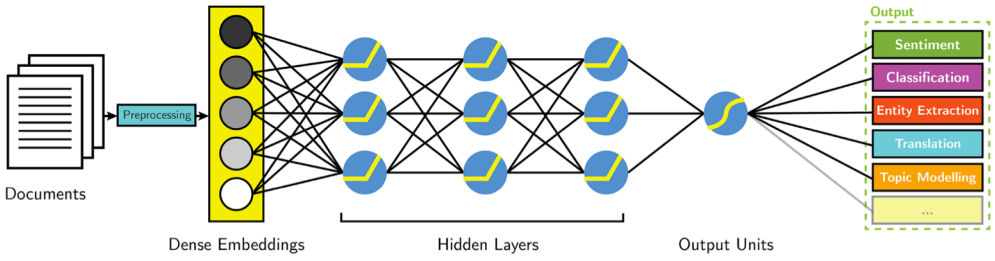


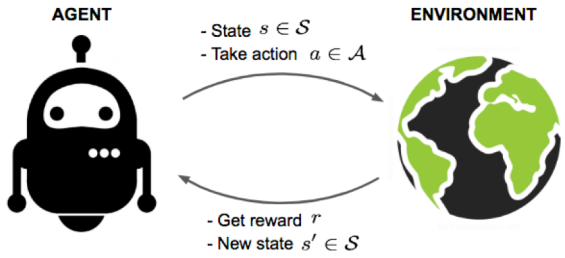
DARPA IDEA Grant, NSF Grant, Best Open-source Tool Awards (IEEE IPDPS, ACM MM, WOSET, CPPConf, ACM DEBS, ACM/IEEE DAC, IEEE/ACM ICCAD, IEEE TCAD, ACM TAU, US Patents, etc.)

MLCraft – Machine Learning at Scale

- **ML system development is very complex**
  - Hard to track experiments
  - Hard to reproduce results
  - Hard to deploy and maintain
- **MLCraft aims to be a unified ML platform**
  - Automate ML workflows and developments
  - Open API and interface to work with existing libraries
  - More robust, predictable, easier to maintain





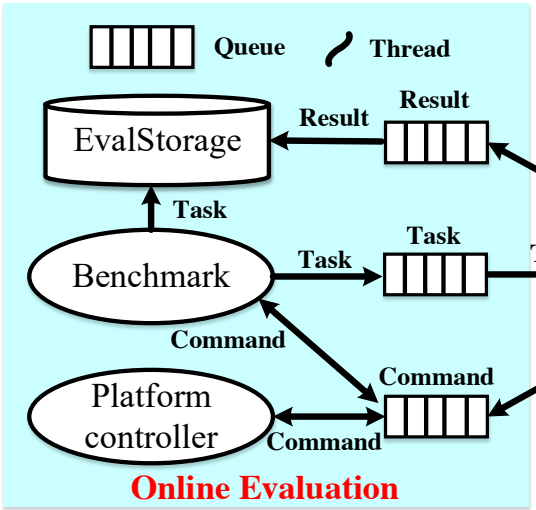


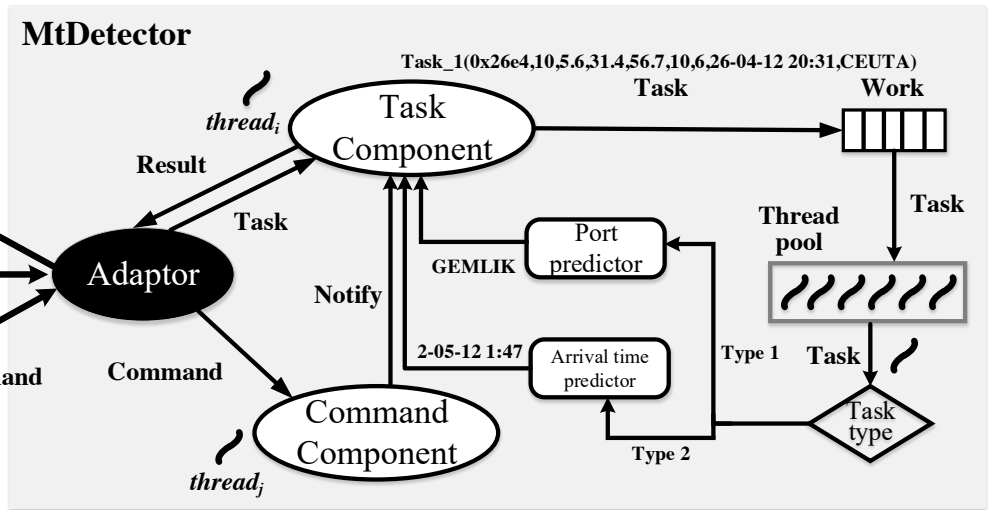
```
dtc::Graph G;

auto src = G.insert<dtc::cell::MnistStreamFeeder>(
  mnist_image_file,
  mnist_label_file,
  [] (Eigen::MatrixXf& images, Eigen::VectorXi& labels) {
    images /= 255.0f;
    return std::make_tuple(images, labels);
  }
);

src.window(1ms).frequency(1000);

auto dnn = G.insert<dtc::cell::Visitor1x1>(
  [] (dtc::ml::DnnClassifier& c) {
    dtc::cout("Creating a dnn classifier [784x30x10]\n").flush();
    c.fully_connected_layer(784, 30, dtc::ml::Activation::RELU)
     .fully_connected_layer(30, 10, dtc::ml::Activation::NONE)
     .optimizer<dtc::ml::AdamOptimizer>();
  },
  [i=0] (dtc::ml::DnnClassifier& c, std::tuple<Eigen::MatrixXf, Eigen::VectorXi>& data) mutab
    auto& [images, labels] = data;
    dtc::cout(
      "Accuracy at training cycle ", i++, " [", images.rows(), " images]: ",
      ((labels-c.infer(images)).array() == 0).count() / static_cast<float>(images.rows()), '\
    ).flush();
    c.train(images, labels, 10, 64, 0.01f, [](){});
  }
);

dnn.in(src.out());

G.container().add(dnn);
G.container().add(src);

dtc::Executor(G).run();
```

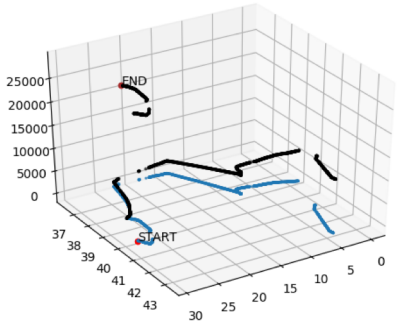
Only **“60-line”** code to automate a distributed ML workflow!

Marine Traffic Prediction at Stream Scale

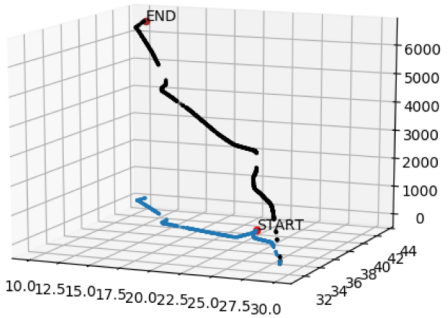
- **MtDetector: A stream-based online ML system**
  - Top-5 in ACM DEBS 2018 Grand Challenge (out of 50+)



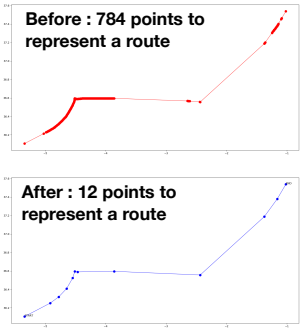


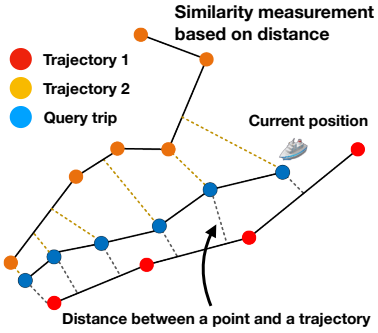


MAE = 120 minutes



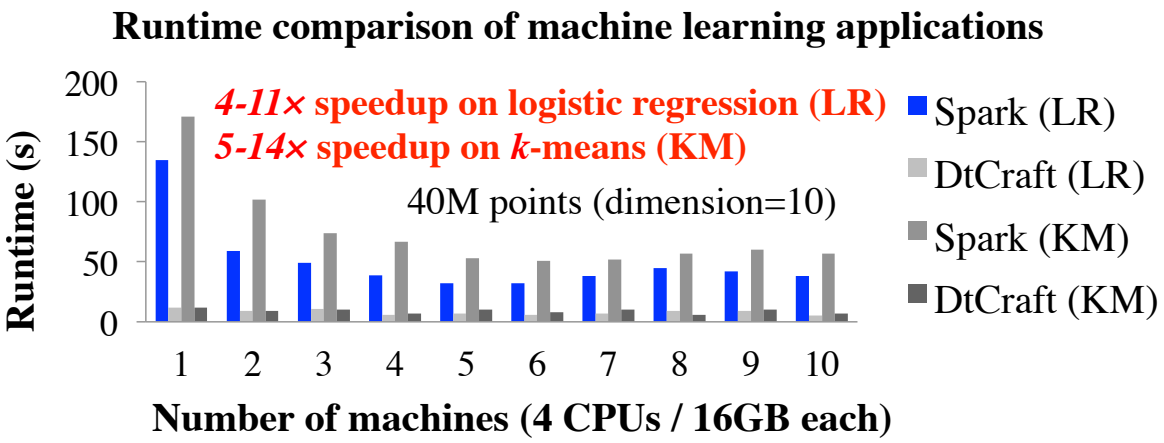
MAE = 267 minutes

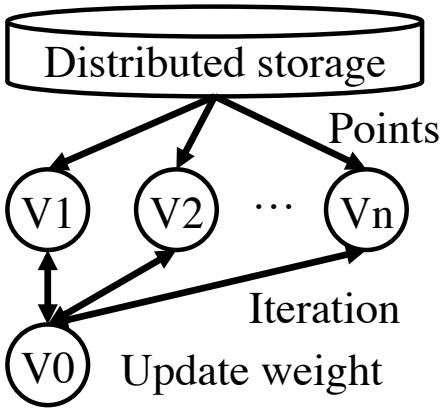




Machine Learning Performance

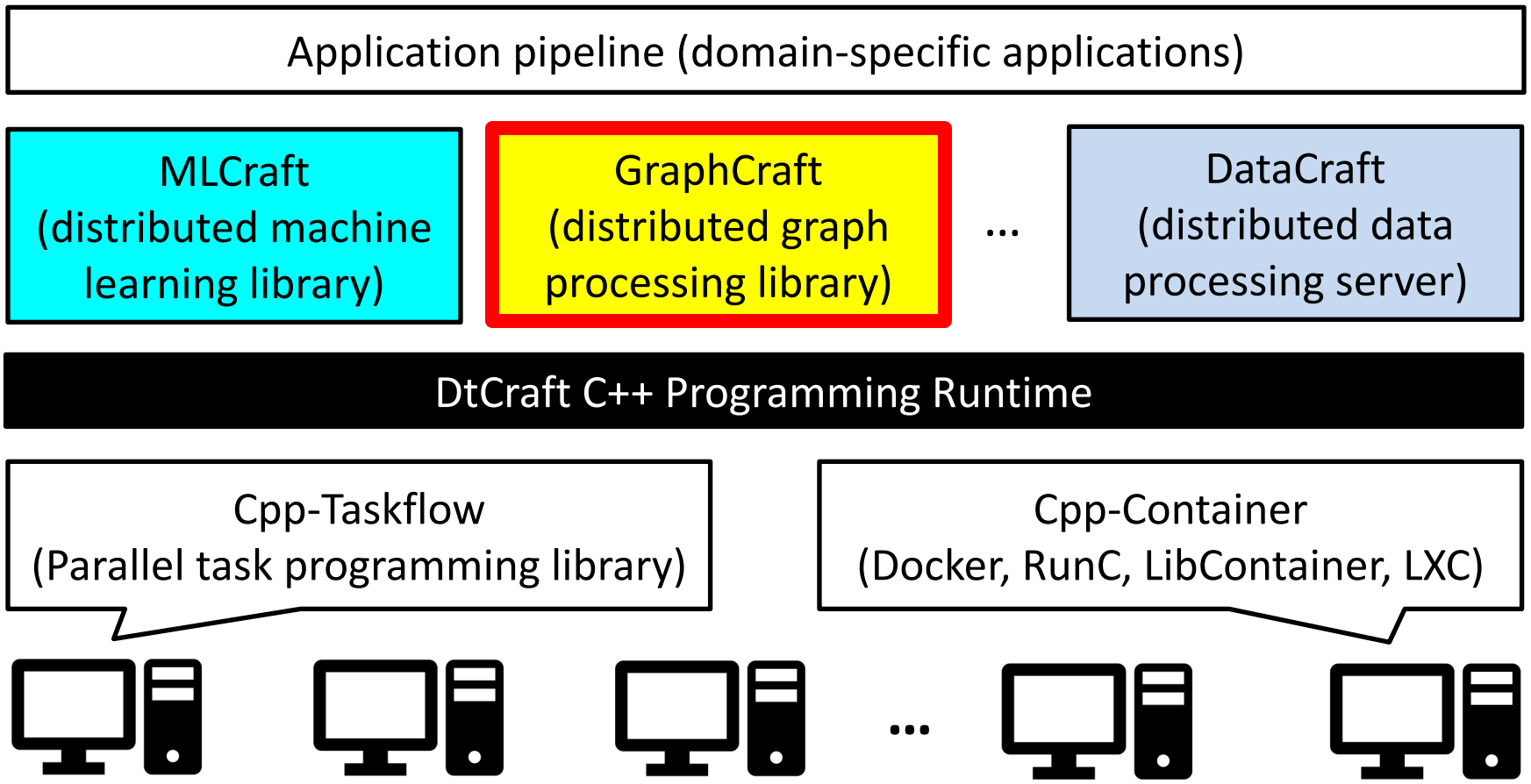
- **Logistics regression and *k*-means clustering**
  - 100 training iterations
- **Compared with Spark ML library**
  - 15x speed-up with 2-5x less memory
  - No extra overhead to cache the data for reuse





Stream graph

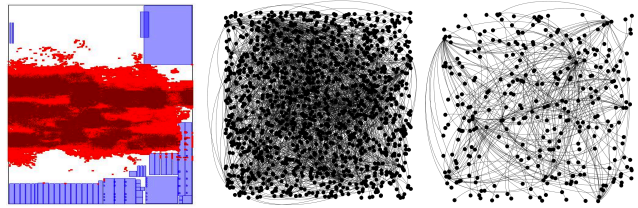
DtCraft to Handle Graph Algorithms

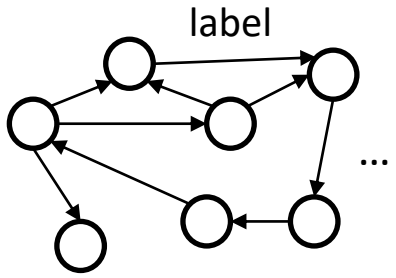


DARPA IDEA Grant, NSF Grant, Best Open-source Tool Awards (IEEE IPDPS, ACM MM, WOSET, CPPConf, ACM DEBS, ACM/IEEE DAC, IEEE/ACM ICCAD, IEEE TCAD, ACM TAU, US Patents, etc.)

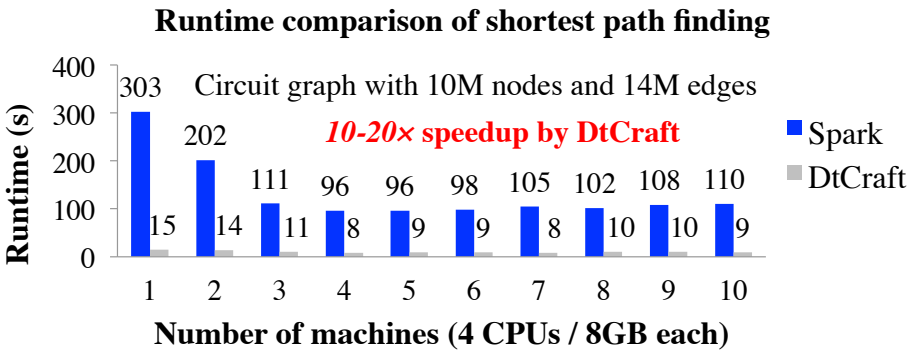
Graph Algorithms Performance

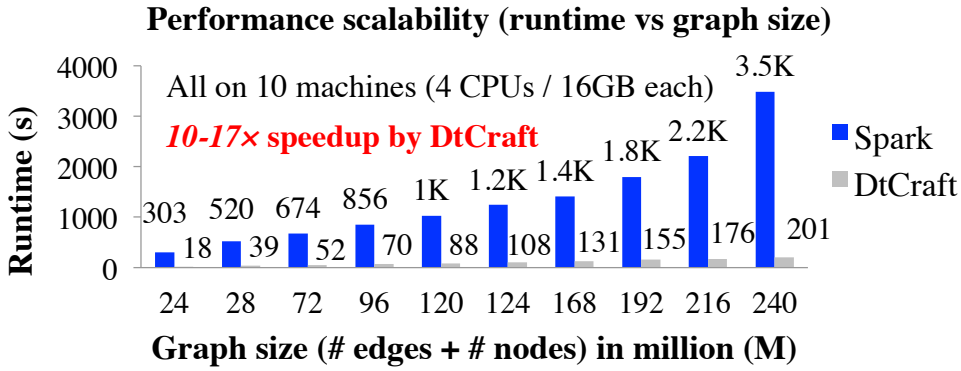
- **Shortest path finding**
  - A circuit graph of 10B transistors
  - Pregel algorithms
- **Compared with Spark GraphX**
  - 10-20x speed-up
  - Better scalability in increasing graph sizes
  - Less communication cost


(a) Circuit (1.01mm²) (b) Graph (3M gates) (c) A signal path

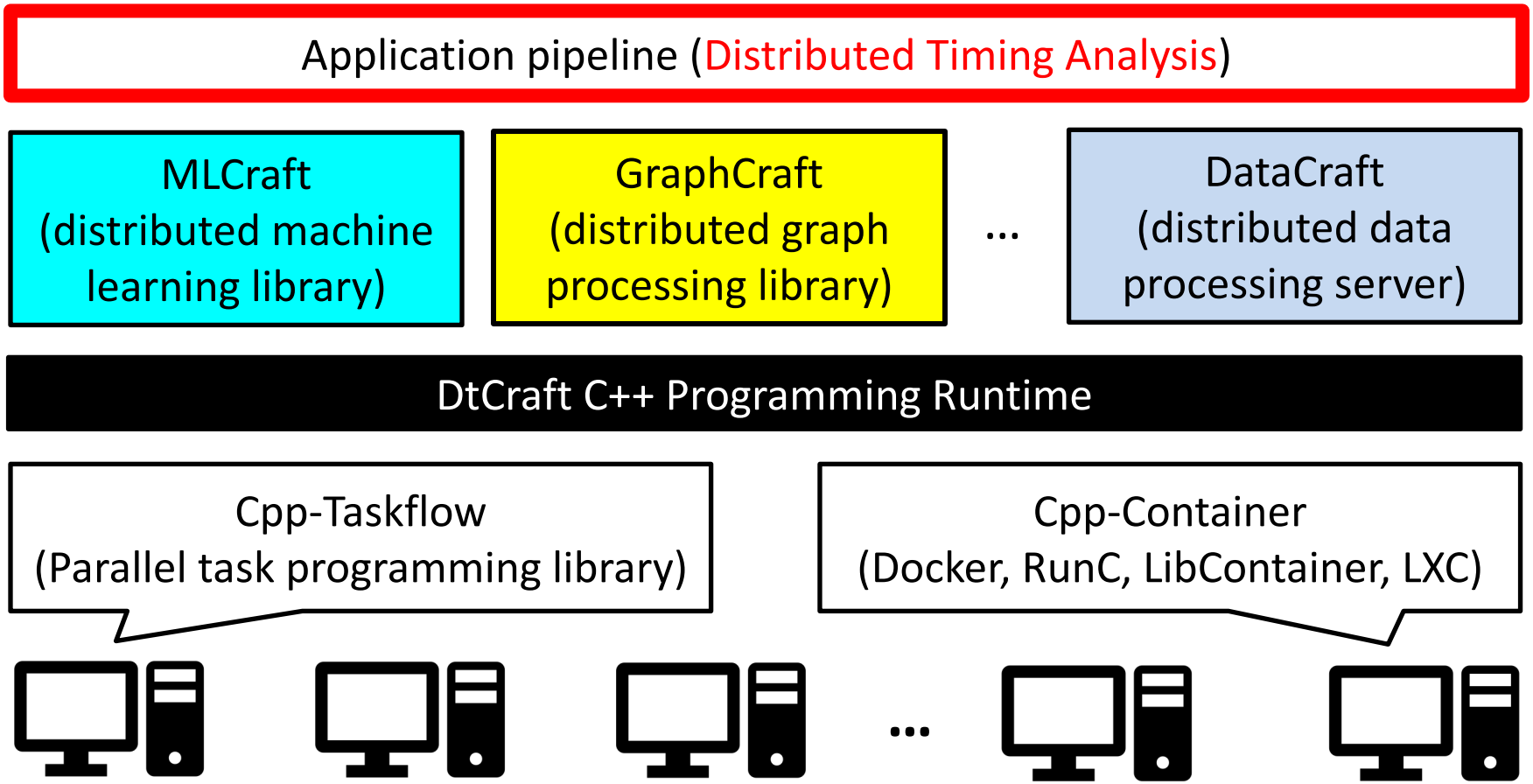


Stream graph





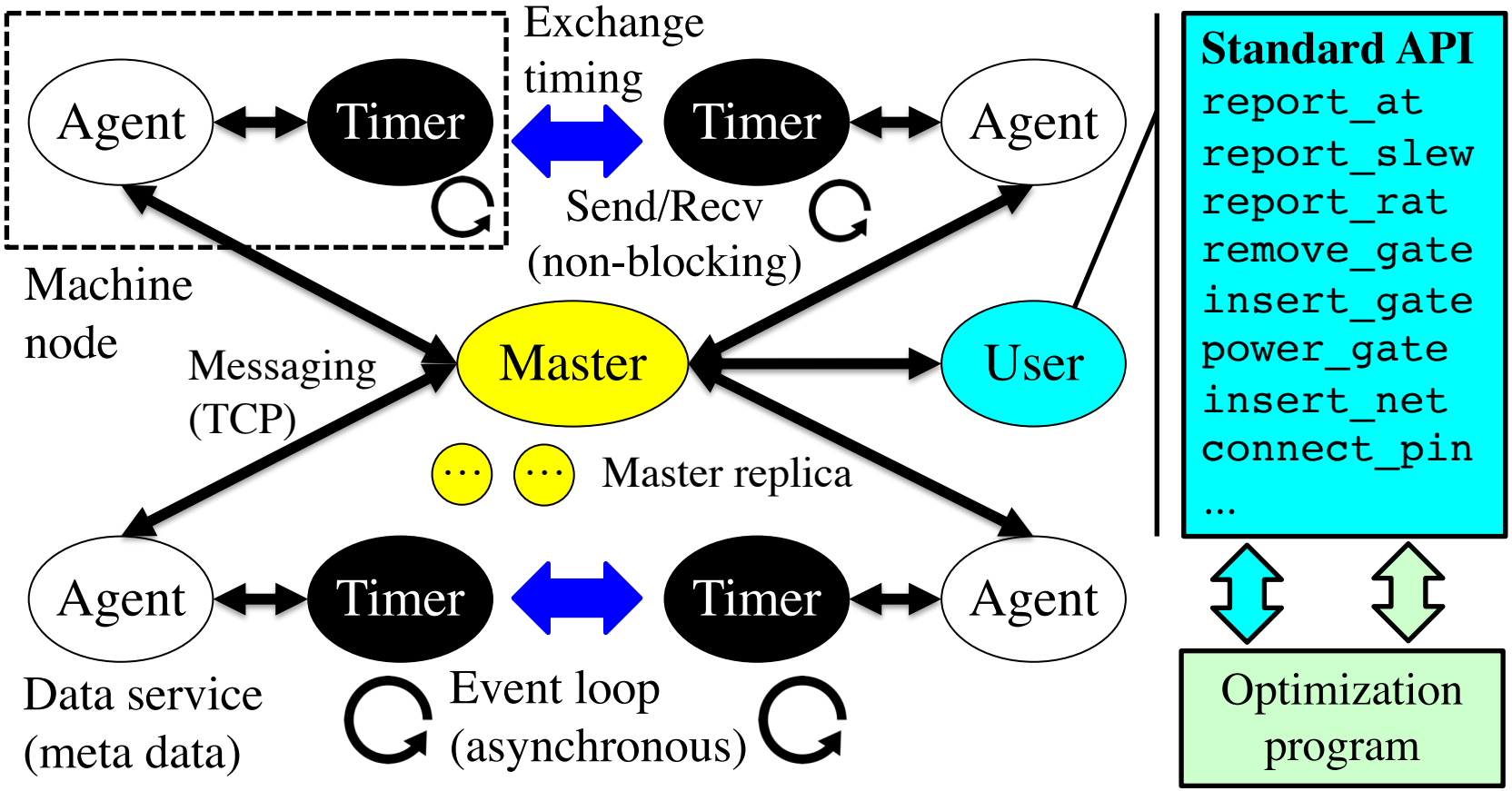
DtCraft to Handle Distributed Timing



DARPA IDEA Grant, NSF Grant, Best Open-source Tool Awards (IEEE IPDPS, ACM MM, WOSET, CPPConf, ACM DEBS, ACM/IEEE DAC, IEEE/ACM ICCAD, IEEE TCAD, ACM TAU, US Patents, etc.)

Distributed Timing Analysis

- Master coordinator and worker (a timer per partition)



Exchange Timing Values

DtCraft

```
// Timing data (early/late rise/fall)
struct Timing {

  string pin;
  array<float, 4> value;

  template <typename T>
  auto archive(T& ar) {
    return ar(pin, value);
  }
};

// Timing path
struct Path {

  float slack;
  vector<string> pins;

  template <typename T>
  auto archive(T& ar) {
    return ar(slack, pins);
  }
};

// Exchange timing through DtCraft stream
stream.on(
  [](Vertex& v, InputStream& is) {
    if(Timing timing; is(timing) != -1) {
      // Call OpenTimer to run incremental timing
    }
  }
);
```

In-context streaming with < 30 lines

Existing framework

Hard-code your message format

```
// OpenTimer.proto
package OpenTimer;

// Message format for timing
message Timing {
  required string pin = 0;
  required float er = 1;
  required float ef = 2;
  required float lr = 3;
  required float lf = 4;
}

// Message format for path
message Path {
  required float slack = 0;
  repeated string pins = 1;
}
```

Many extra stuff ☹

Extra.pb.h

Extra.pb.cpp

…

Source.cpp

Google Protocol Buffer (open-source compiler)

C++/Java/Python source code generator

.cpp/.h class methods

ParseFromArray(void*, size_t)

SerializeToArray(void*, size_t)

Message wrapper

Derived packet struct

header_t header

void* buffer

Out-of-context streaming takes > 300 lines

Deploy the Distributed Timer in One line

DtCraft

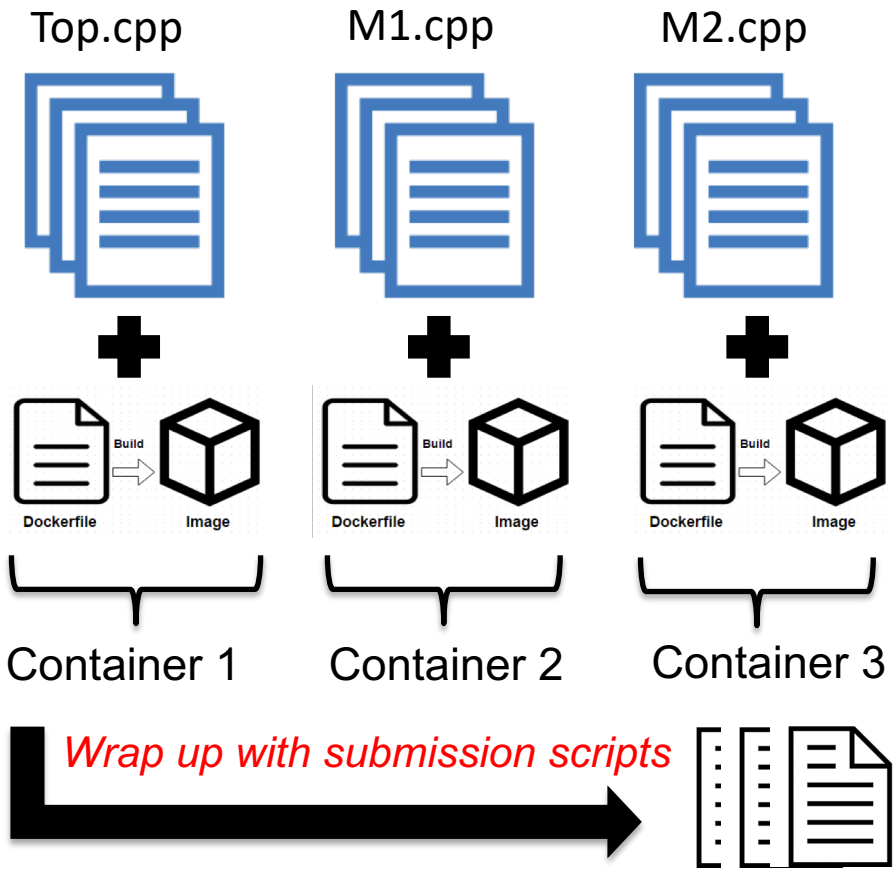
```
// Create a timer vertex for Top
auto Top = G.Vertex().on(
  [=] () {
    OpenTimer timer ("Top.v");
  }
);

// Create a timer vertex for Macro 1
auto M1 = G.Vertex().on(
  [=] () {
    OpenTimer timer ("M1.v");
  }
);

// Create a timer vertex for Macro 2
auto M1 = G.Vertex().on(
  [=] () {
    OpenTimer timer ("M2.v");
  }
);

// Create streams ...
...

// Distribute timers to machines.
G.container().add(Top).num_cpus(4).memory_(4_GB);
G.container().add(M1).num_cpus(1).memory(8_GB);
G.container().add(M2).num_cpus(2).memory(6_GB);
```

Only three lines for resource control in Linux container

~$ ./submit –master=127.0.0.1 binary

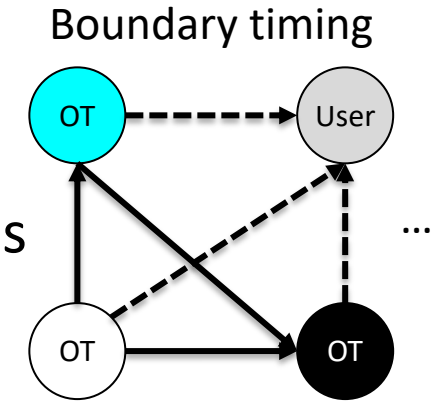
Existing framework

Duplicate the code for each partition

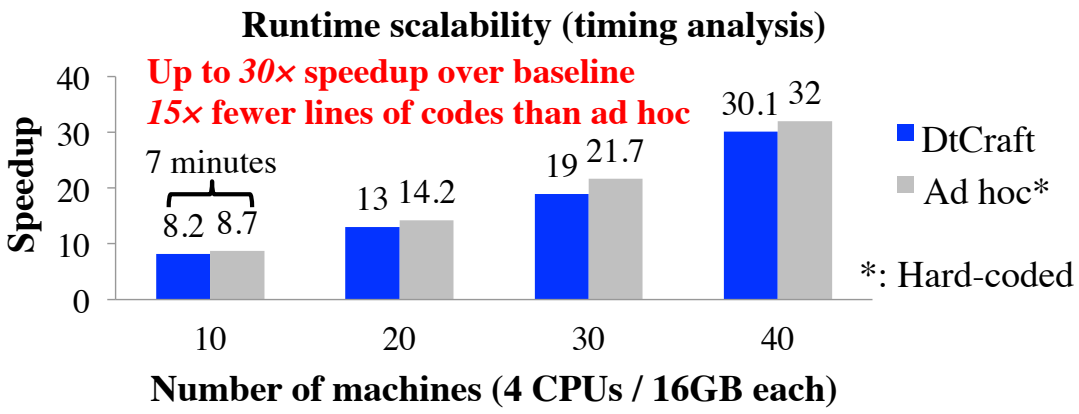


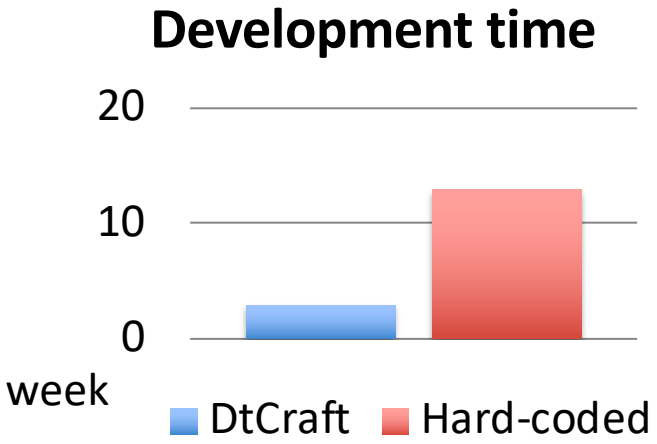
Distributed Timing Performance

- **IBM design of 250 partitions**
  - OpenTimer as the timing analyzer
  - On a cluster of 40 Amazon EC2 instances
- **Compared with hard-coded solution**
  - 15x fewer lines of code
  - Only 7% performance loss

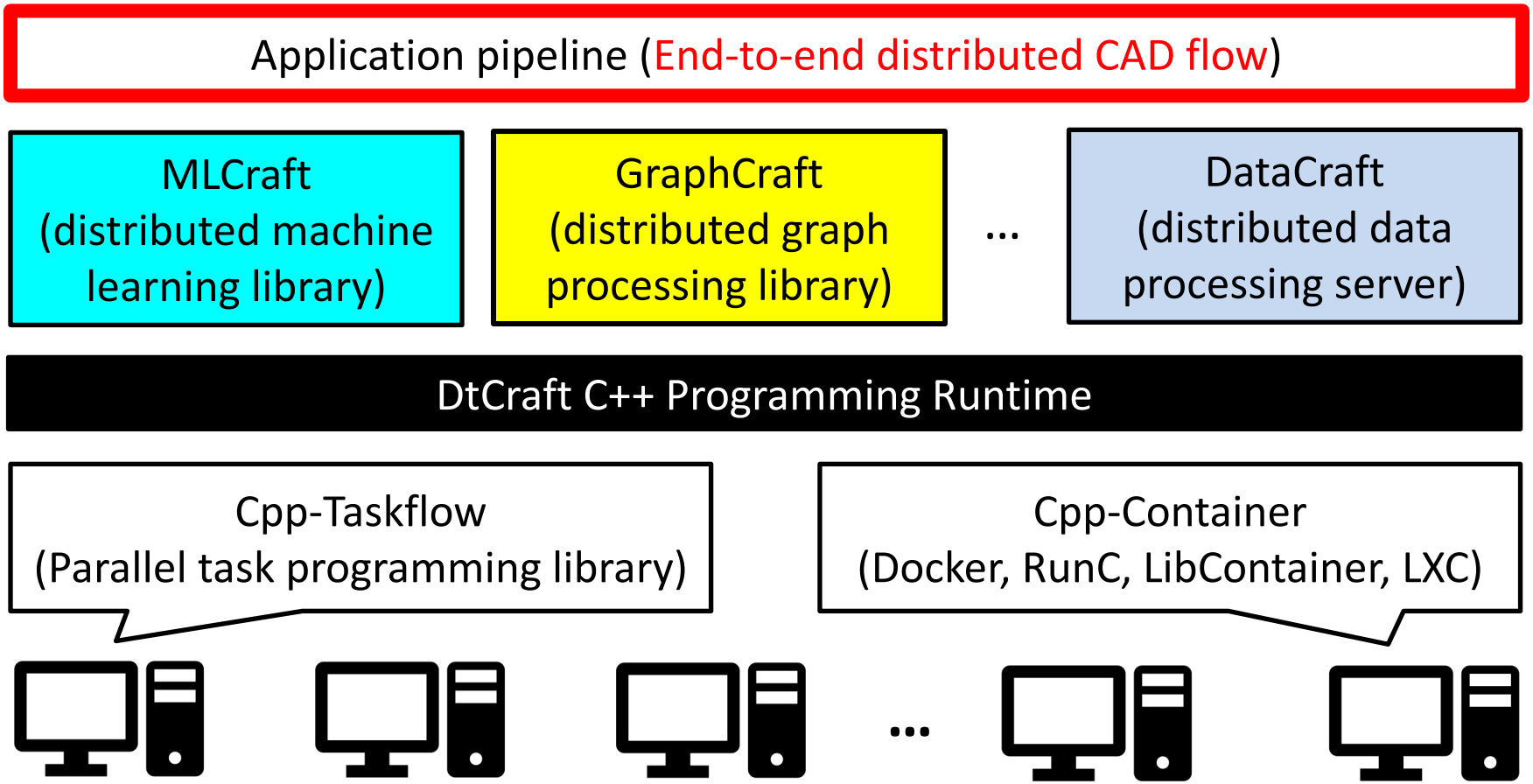


Stream graph
(250 IBM design partitions)





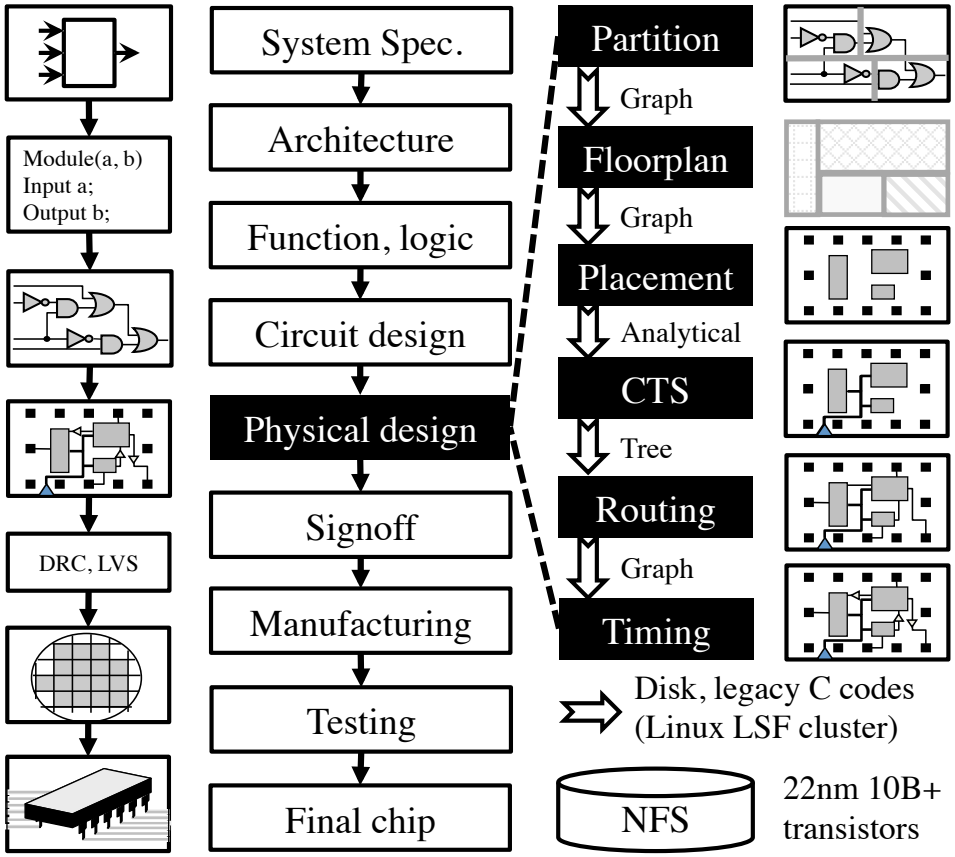
DtCraft to Handle a Layout Generator



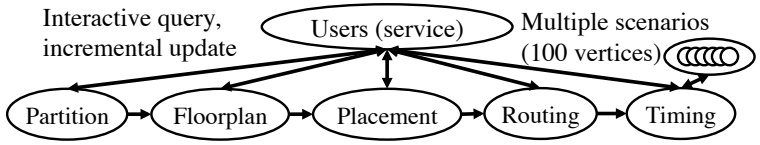
DARPA IDEA Grant, NSF Grant, Best Open-source Tool Awards (IEEE IPDPS, ACM MM, WOSET, CPPConf, ACM DEBS, ACM/IEEE DAC, IEEE/ACM ICCAD, IEEE TCAD, ACM TAU, US Patents, etc.)

A Fully Automated Layout Generator

❑ DARPA's vision: A GCC-like silicon compiler



Stream graph
(106 vertices and 220 streams)



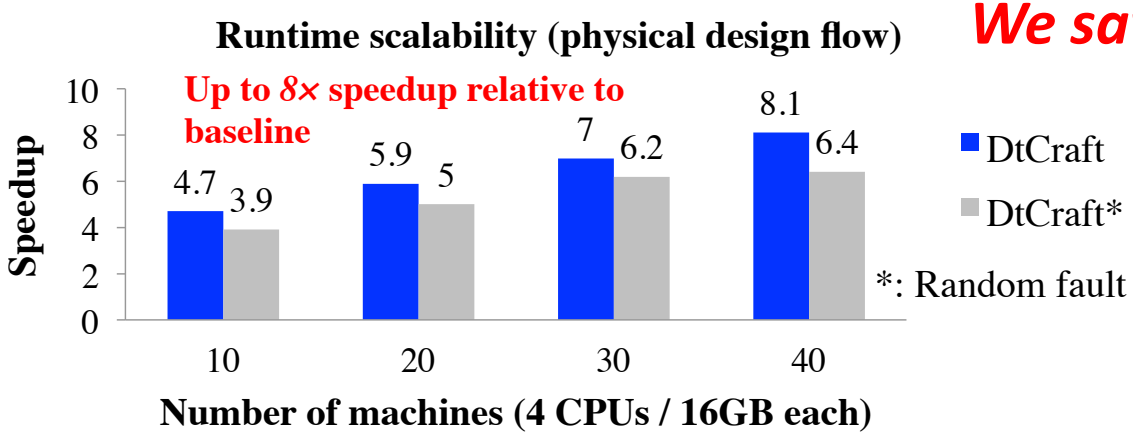
- ❑ **Tool collections**
  - ❑ Partitioner & Floorplanner
  - ❑ Placer & router
  - ❑ Timer
  - ❑ …
- ❑ **IDEA/POSH end state …**

```
$ git clone https://github.com/darpa/idea
$ git clone https://github.com/darpa/posh
$ cd posh
$ make soc42
```

T.-W. Huang, et al, "DtCraft: A high-performance distributed execution engine at scale," IEEE TCAD, 2018

Performance of Distributed Integration

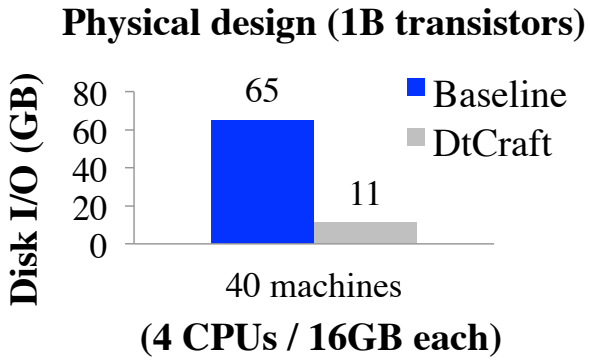
- Compared with existing "script-based" solutions
  - Run on 40 Amazon EC2 nodes

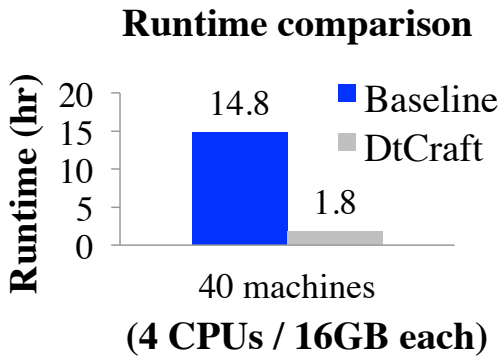


We saved 20x code in general!

* Up to 8x speed-up compared to the baseline

* Design completion time is reduced from by **13 hours**

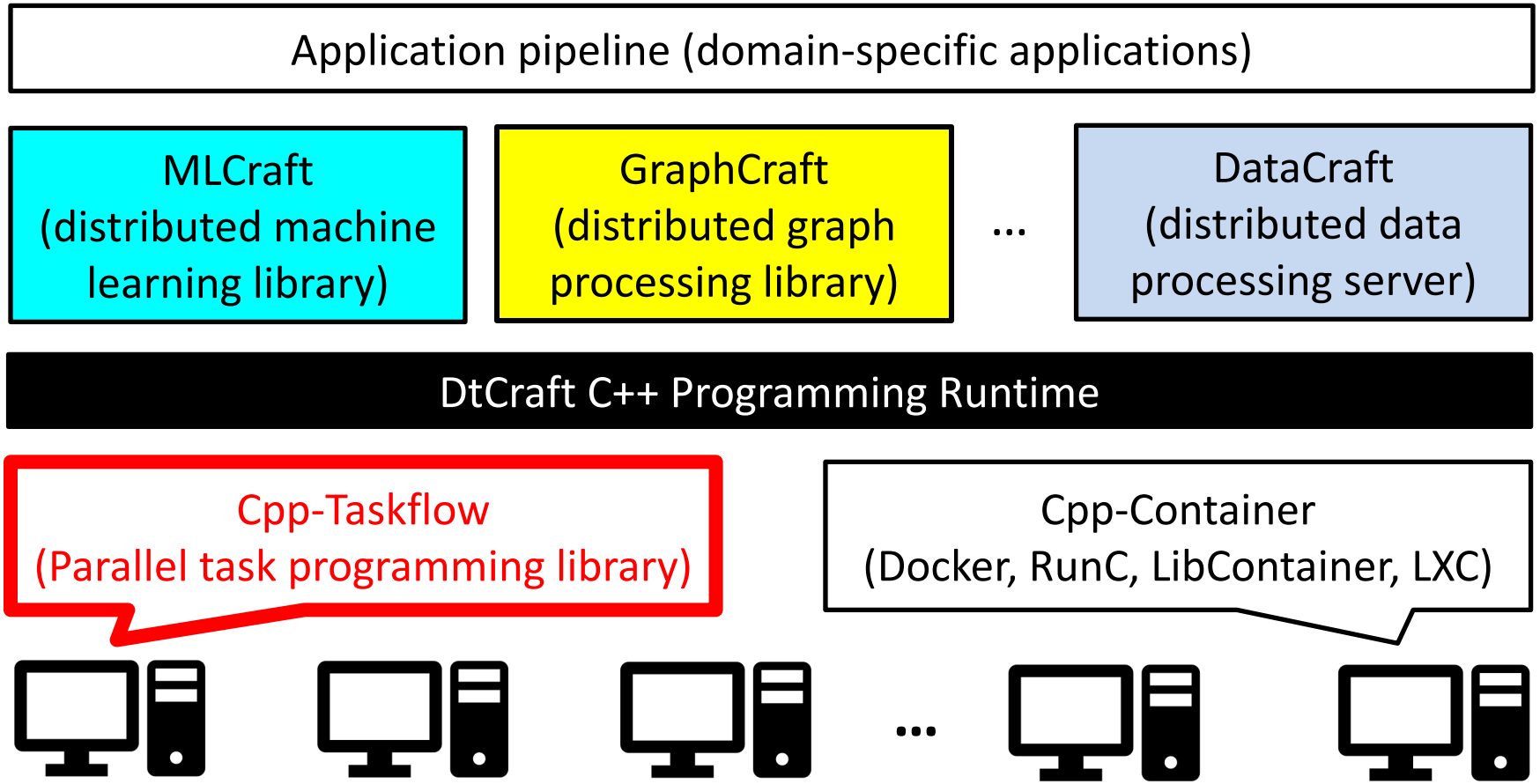




* In-memory streaming reduced the I/O overhead by **~55GB**

* Less I/O overhead translates to faster runtime

DtCraft to Handle Low-level Concurrency



DARPA IDEA Grant, NSF Grant, Best Open-source Tool Awards (IEEE IPDPS, ACM MM, WOSET, CPPConf, ACM DEBS, ACM/IEEE DAC, IEEE/ACM ICCAD, IEEE TCAD, ACM TAU, US Patents, etc.)

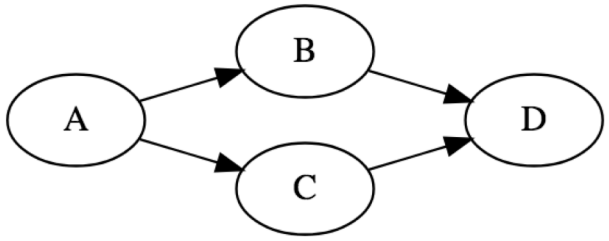
Cpp-Taskflow: Parallel Programming Library

*We aim to help developers **quickly** write **efficient** parallel programs using task-based approaches with **Modern C++***

Cpp-Taskflow's Project Mantra

- **We want parallel code**
  - Simple
  - Expressive
  - Transparent
- **We don't want hard details**
  - Explicit thread management
  - Difficult lock controls
  - Daunting class declaration

```
tf::Taskflow tf;

auto [A, B, C, D] = tf.silent_emplace(
  [] () { std::cout << "Task A\n"; },
  [] () { std::cout << "Task B\n"; },
  [] () { std::cout << "Task C\n"; },
  [] () { std::cout << "Task D\n"; }
);

A.precede(B, C);   // A runs before B and C
B.precede(D);      // B runs before D
C.precede(D);      // C runs before D

tf.wait_for_all();  // block until finish
```

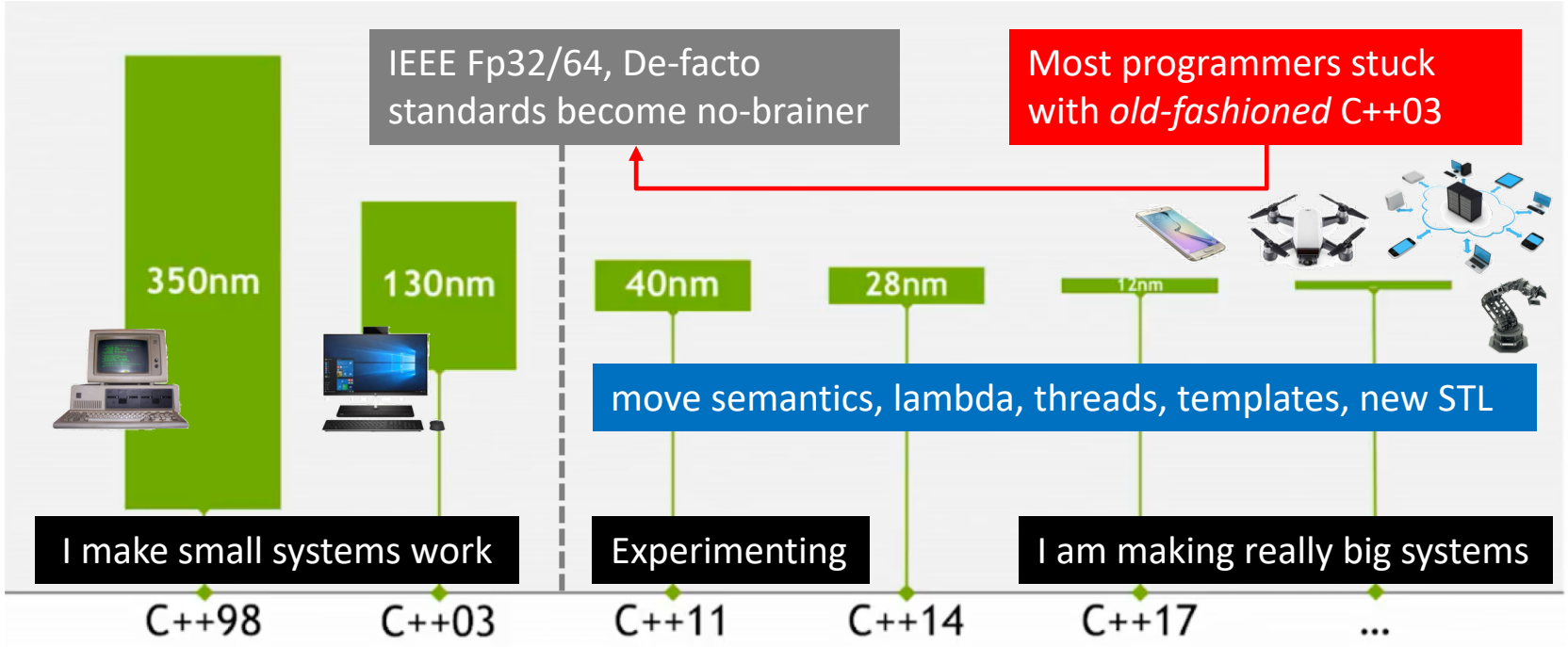


T.-W. Huang, et al, "Cpp-Taskflow: Fast Parallel Task-based Programming using Modern C++," IEEE IPDPS, 2019

*Only **10 lines** of code to enable parallel task execution!*

“Modern C++” Enables New Technology

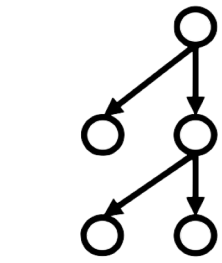
- If you were able to tape out C++ …



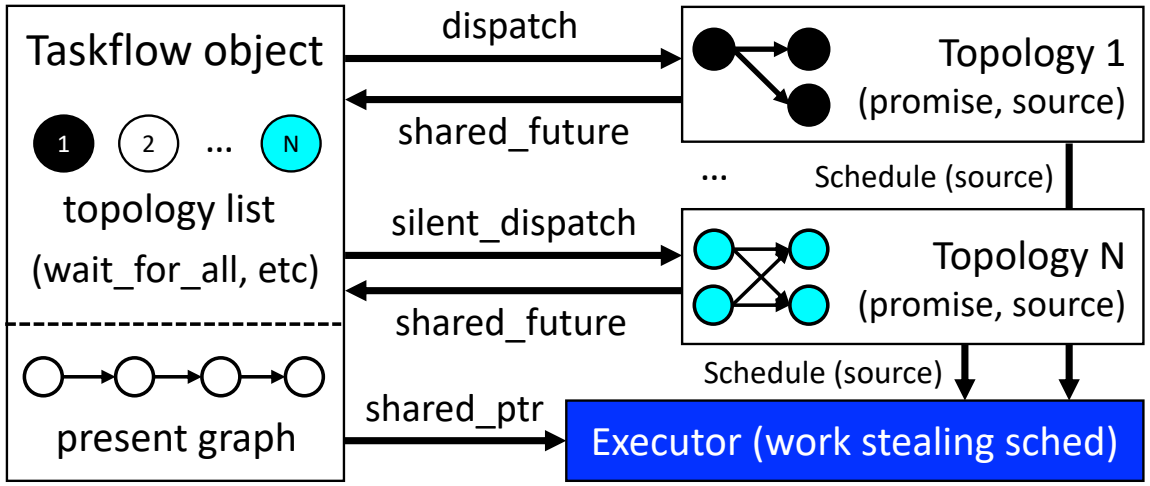
- It’s much more than just being modern
  - Must “rethink” the way we used to design a program
  - Achieve the performance previously not possible

Implementation Details of Cpp-Taskflow

- **A new parallel task programming model**
  - Simple yet efficient
- **Unified interface for both static and dynamic tasking**
  - No need to learn a different API set
- **Pluggable scheduler and executor interface**
  - Modular and customizable
  - Sharable

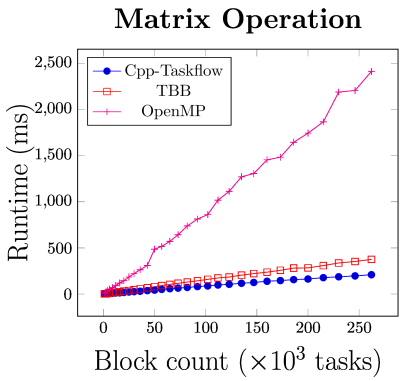


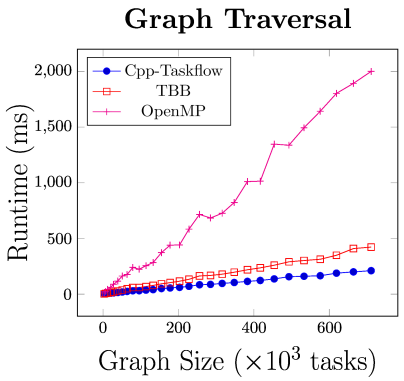
Cpp-Taskflow

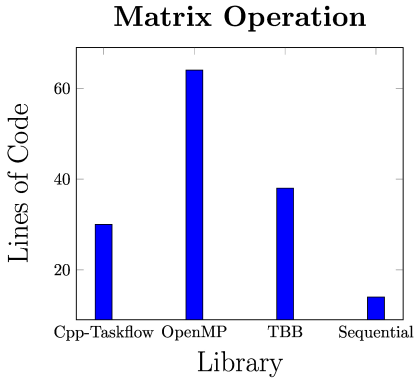


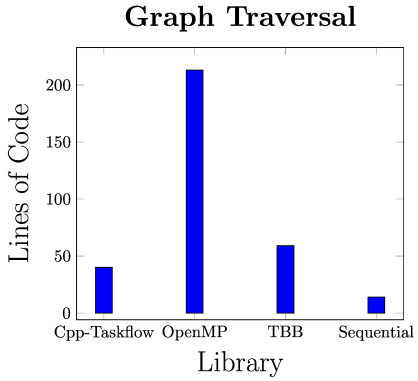
Cpp-Taskflow's Performance

- **Compared with OpenMP and Intel TBB**
  - Faster, more scalable, and fewer lines of code



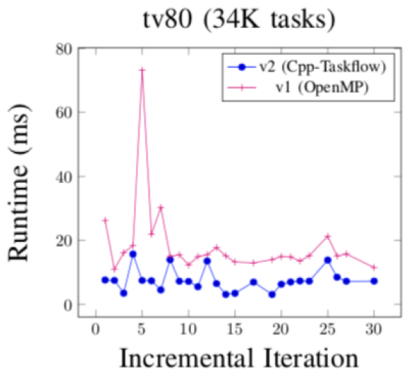


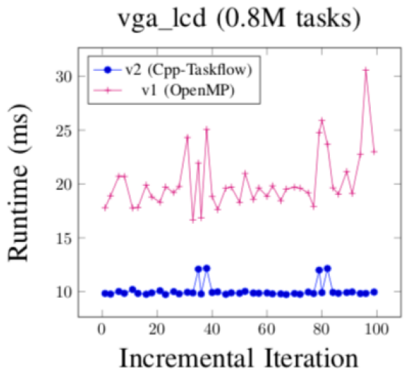




- **Large VLSI timing analysis (OpenMP vs Cpp-Taskflow)**
  - 2x speed-up over OpenMP
  - 9123 vs 4482 lines of code

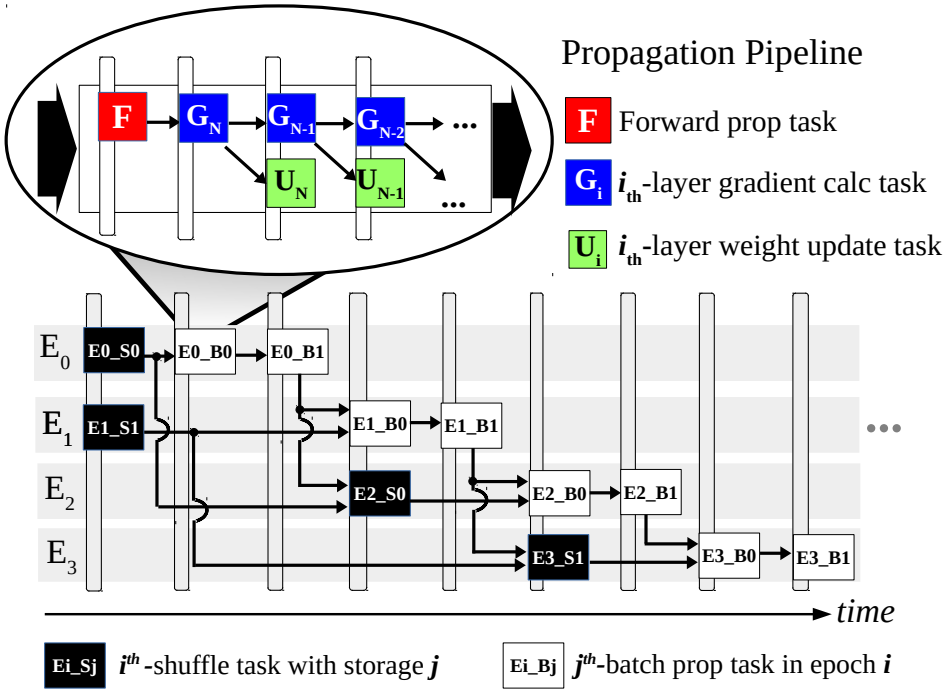
Cost to develop is $275K with OpenMP vs $130K with Cpp-Taskflow!
(https://dwheeler.com/sloccount/)





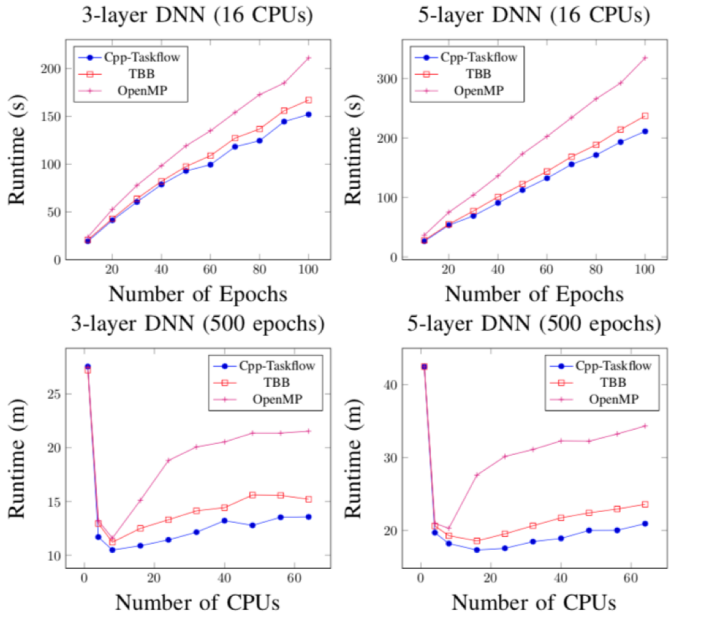
Simplify the Development of Parallel ML

- **"OMG, never do so in TensorFlow,"** *G____ Research*
  - RunQueue, Python threads are nightmare in TensorFlow
- **Cpp-Taskflow makes developers' lives easier**
  - 2x fewer LoC and 2x faster



Cpp-Taskflow			OpenMP			TBB			Sequential		
LOC	CC	T	LOC	CC	T	LOC	CC	T	LOC	CC	T
59	11	3	162	23	9	90	12	3	33	9	2

CC: cyclomatic complexity of the implementation
T: development time (in hours) by an experienced programmer



What Cpp-Taskflow Users Say …

*"**Cpp-Taskflow is the cleanest Task API I have ever seen,**"* *Damien Hocking*

*"**Cpp-Taskflow has a very simple and elegant tasking interface; the performance also scales very well,**" Totalgee*

*"**Best poster award for open-source parallel programming library,**" Cpp-Conference (voted by 1000+ developers)*

People are using Cpp-Taskflow to speed up TensorFlow's kernel!

lifeiteng commented on Dec 11, 2018

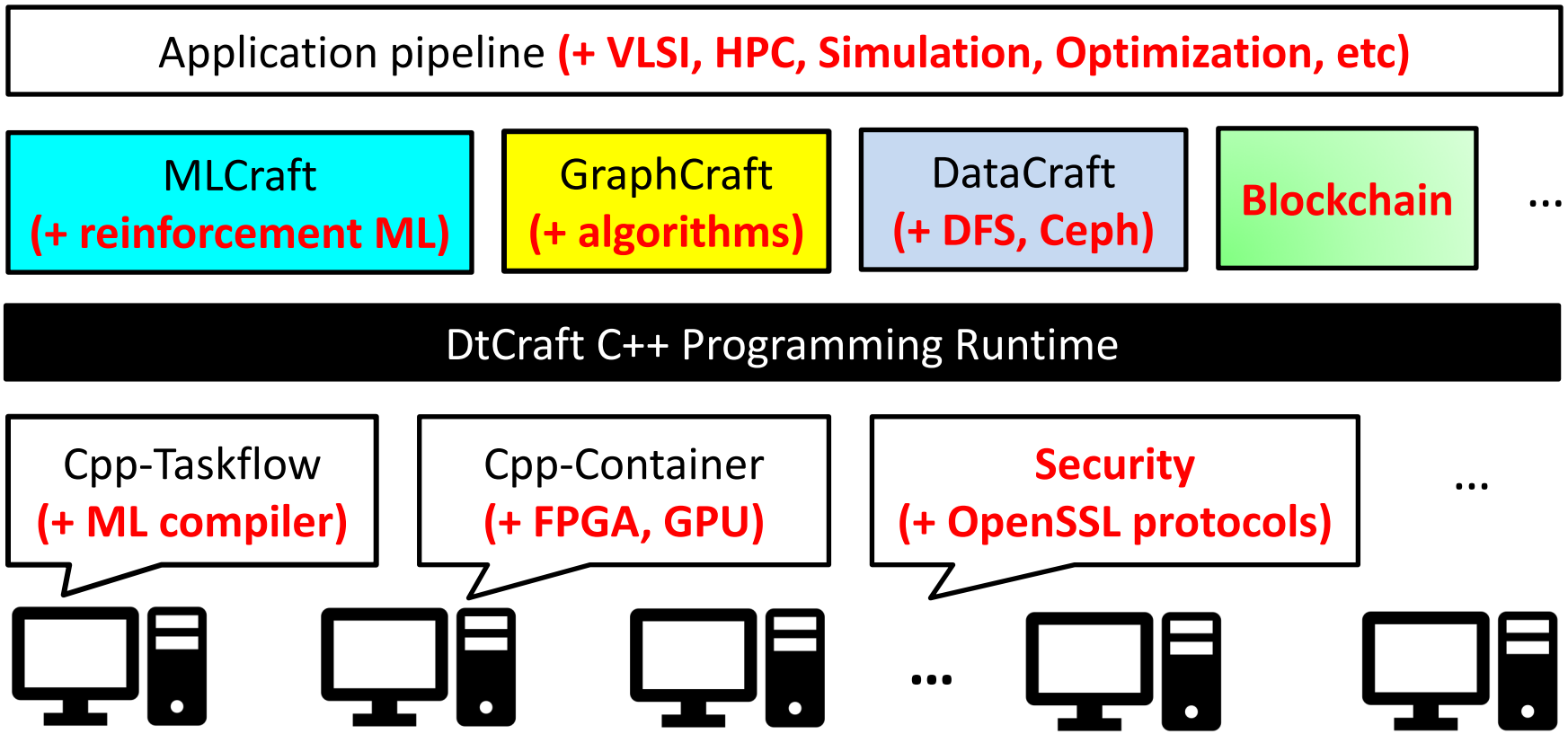
Is your feature request related to a problem? Please describe.
I want to use TaskFlow in a C++ project which depends on TensorFlow. When compiling the Op & Kernel with `c++17`, there is a problem tensorflow/tensorflow#23561

Outline

- **My early PhD research**
  - OpenTimer: A VLSI timing analysis tool
  - Express parallelism in the right way
- **A general-purpose distributed programming system**
  - DtCraft and its ecosystem
  - Machine learning, graph algorithms, chip design
  - Cpp-Taskflow: Modern C++ parallel task programming
- **Conclusion and future work**

Future Work (marked red)

- **Enhance both app- and system-level capabilities**
  - New workloads, new libraries, new API, etc.



Collaboration Plan

- **With faculty members working on (not limited to):**
  - HPC, FPGA, networking, machine learning, AI, computer systems, VLSI, OS virtualization, etc.
- **With students (anyone interested in computing!)**
  - Undergraduate and graduate research advising
  - Existing courses (EE, CE, CS)
    - Intro, VLSI, Digital Design, Architecture, Programming, OS, etc.
  - *New course ideas (at all levels)* 

    - Big-data cloud computing, scalable AI systems, virtualization, modern C++ programming, principles of open source, etc.
  - Professional activities (contests, conferences, etc.)
    - Programming contests, research competition, conferences, etc.
- **PI of DARPA research grant "OpenTimer and DtCraft"**
  - Brings collaborations with faculty members & students

Acknowledgment (Users & Sponsors)