近未来的並列 LL

@mootoh

@mootoh

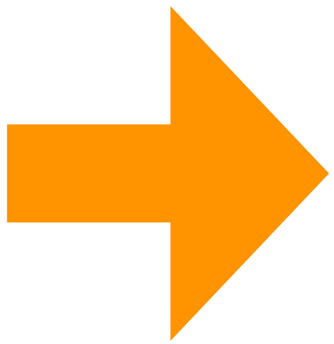
高山 征大

並歴 5年

最近は 研究者：エンジニア ＝ 7:3

http://deadbeaf.org

個人の意見、見解です



Molatomium

CELL REGZA

並カン

http://bit.ly/namikan

並カン

並列プログラミングカンファレンス

日時 / DATE : **2010/01/31 13:00 to 18:00**

このイベントは終了しました

このイベントの参加希望者 **105** / 120 人

参加者105人 / キャンセル42人

■ 参加者 (105 人)

- mootoh : 主催者
- takkanm : 2get.
- yad-EL : まけた
- wraith13 : ｜д´)ｼﾞｰｯ
- negaton : みかんと聞いて。並列化…避けては通れないですよね。
- r153 : 参加しまーす
- kawanishi : すみません。あまり話せる内容ないですが……。最近勉強中です。
- cesare : 参加しまする。

はじめに	mootoh
並列プログラミングの入門＆おさらい的な話	wraith13
ローレイヤーでの並列処理の設計	goyoki
STM	hayamiz
並列 HPU 言語 MUDA	syoyo
マルチコア時代の Lock-free 入門	yamasa
Haskell 周り	shelarcy
その他	???

目的だったもの

- 研究者だけじゃなくて
- ふつうのエンジニアも並列処理を

目的だったもの

- 研究者だけじゃなくて
- ふつうのエンジニアも並列処理を

100人!

目的だったもの

- 研究者だけじゃなくて
- ふつうのエンジニアも並列処理を

100人!

そして LL tiger

近未来的並列 LL

そもそも

なぜ並列なのか

タダ飯
終了の
お知らせ

\(^o^)/

\(^o^)/

\(^o^)/

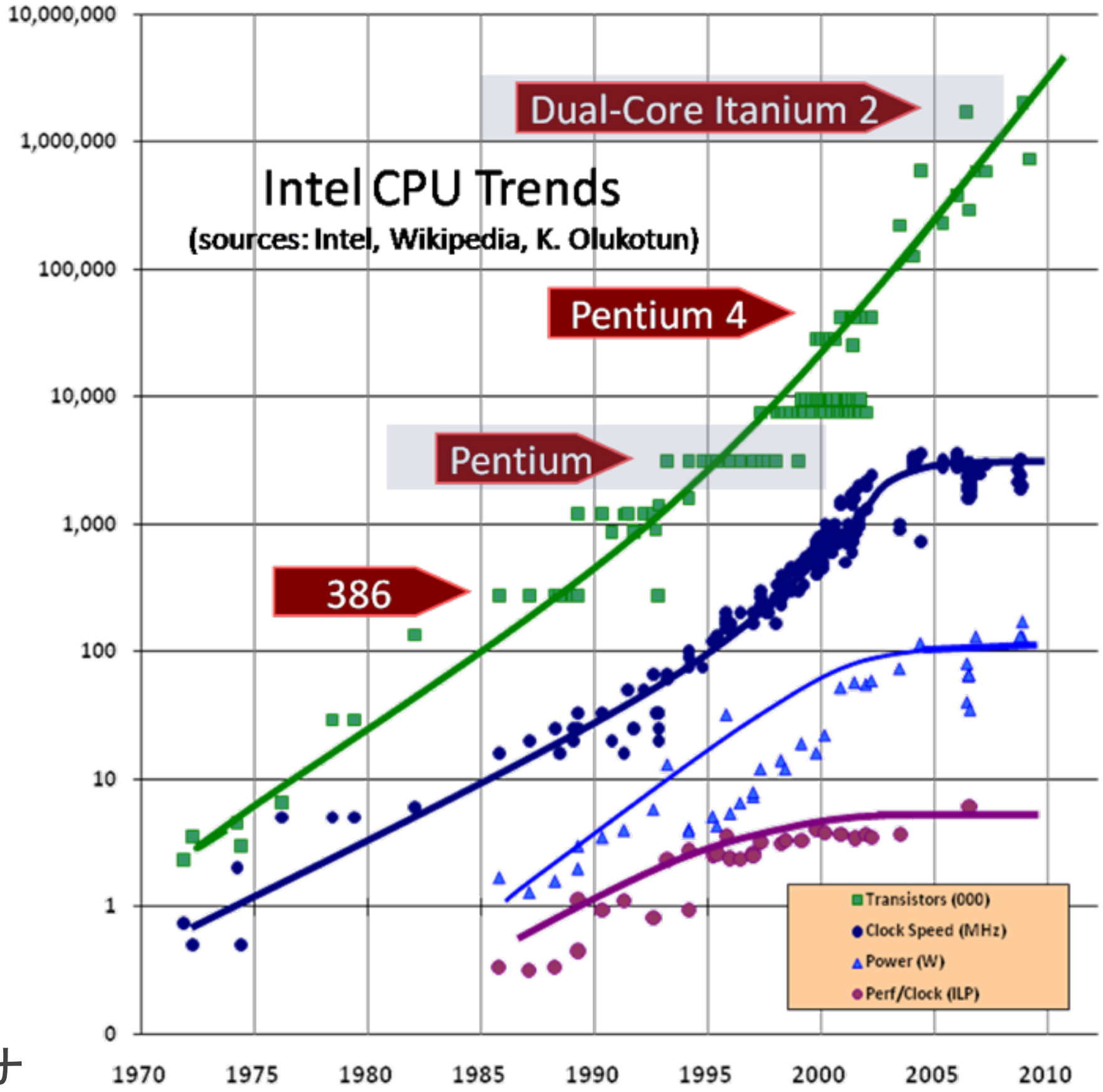
\(^o^)/

\(^o^)/

\(^o^)/

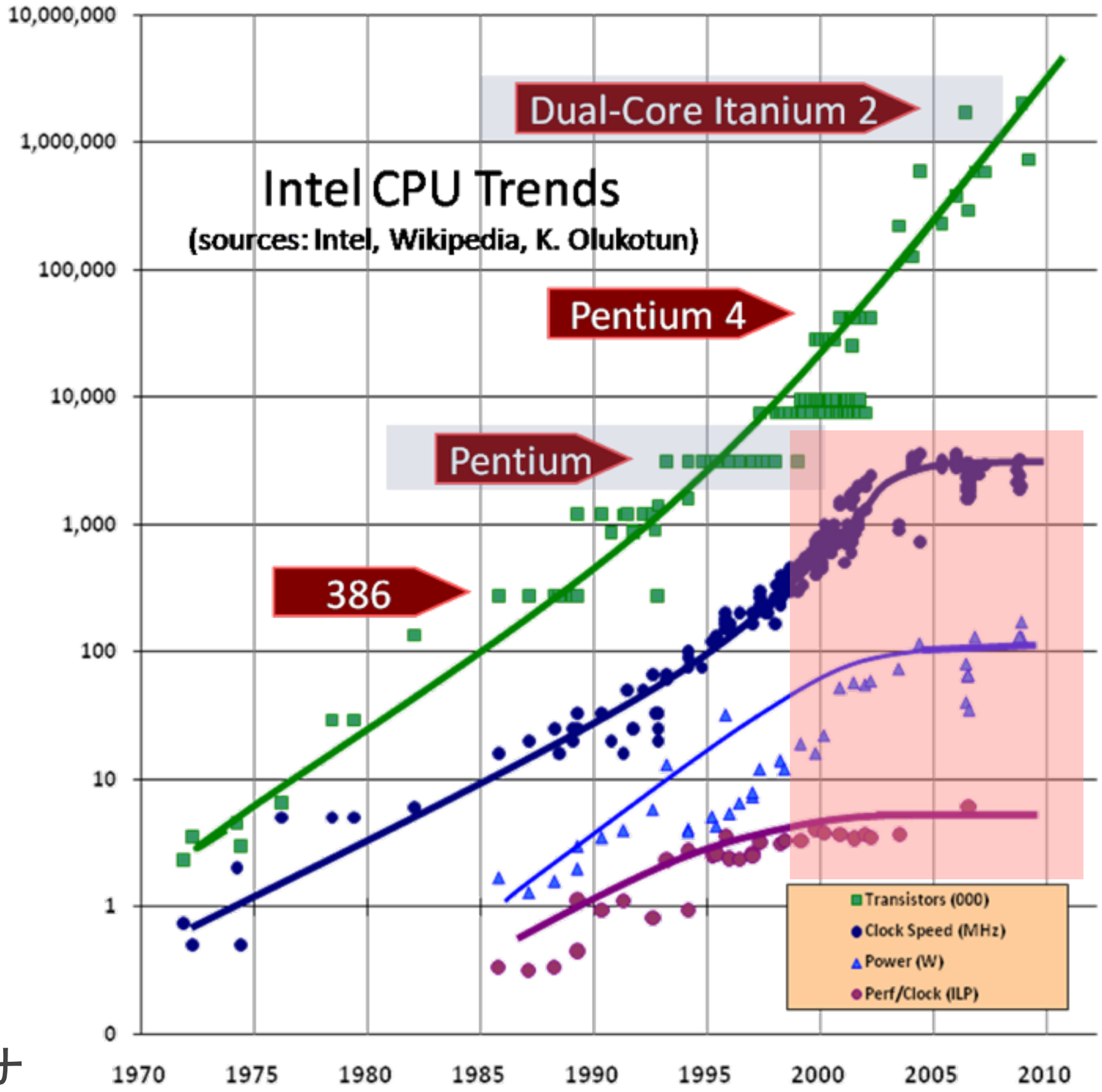
\(^o^)/

タダ飯
終了の
お知らせ



"The Free Lunch is Over", by Herb Sutter
http://www.gotw.ca/publications/concurrency-ddj.htm

タダ飯
終了の
お知らせ



"The Free Lunch is Over", by Herb Sutter
http://www.gotw.ca/publications/concurrency-ddj.htm

どういうことか

- これまで
  - 新しい CPU 買ってくれば勝手に速くなった
- これから (マルチコア時代)
  - そのままじゃ速くならない ＼(^o^)／
  - 並列に書くしかない

マルチコアが　あらわれた！

そもそも速くする

必要があるの?

ここで
派手なデモ

低レイテンシー

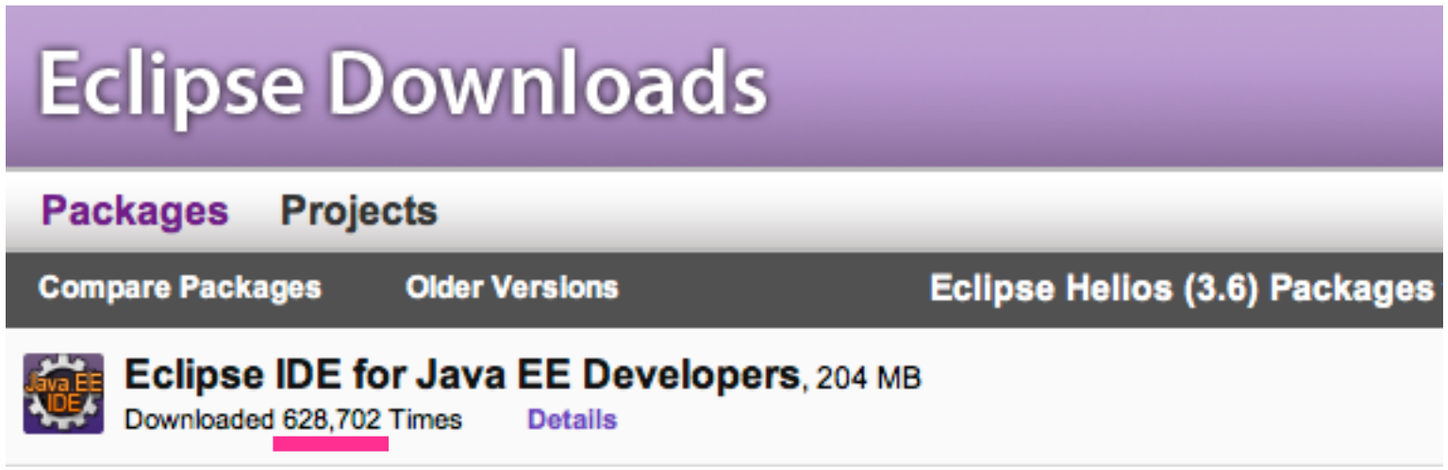
ソフトウェア

待ち時間

- 例: Eclipse.app を起動する時間
  - on iMac Core 2 Duo 3GHz 4GB RAM
  - 初回: 33s, 2回目以降: 7s

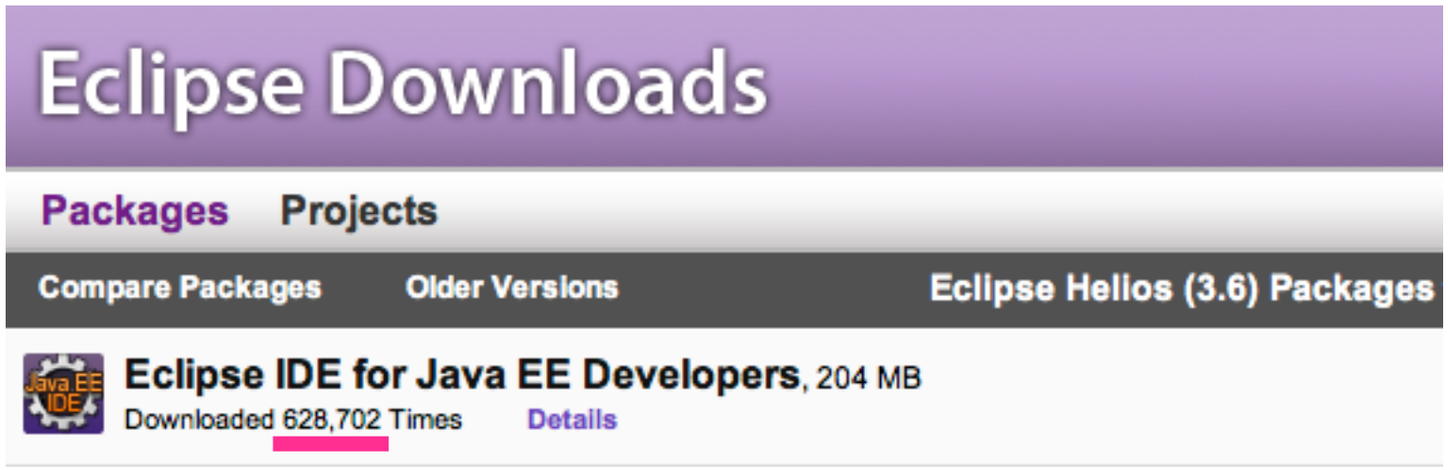
待ち時間

- 例: Eclipse.app を起動する時間
  - on iMac Core 2 Duo 3GHz 4GB RAM
  - 初回: 33s, 2回目以降: 7s



待ち時間

- 例: Eclipse.app を起動する時間
  - on iMac Core 2 Duo 3GHz 4GB RAM
  - 初回: 33s, 2回目以降: 7s



(628,702人 * 33s * 365日) / (60 * 60 * 24 * 30)

=> 2,921 人月/年

待ち時間を減らす

- 21世紀: 時間が最も貴重な資源
- 待たせない
  - 国民総幸福量 (GNH) が向上
- モバイルではより重要

待ち時間を減らす

- 21世紀: 時間が最も貴重な資源
- 待たせない
  - 国民総幸福量 (GNH) が向上
- モバイルではより重要

国民総幸福量

国民総幸福量（こくみんそうこうふくりょう、英: Gross National Happiness, GNH）または**国民総幸福感**（こくみんそうこうふくかん）とは、1972年に、ブータン国王ジグミ・シンゲ・ワンチュクが提唱した「国民全体の幸福度」を示す"尺度"である。国民総生産 (Gross National Product, GNP) で示されるような、金銭的・物質的豊かさを目指すのではなく、精神的な豊かさ、つまり幸福を目指すべきだとする考えから生まれたものである。現在、ブータン政府は国民総幸福量の増加を政策の中心としている。

背景まとめ

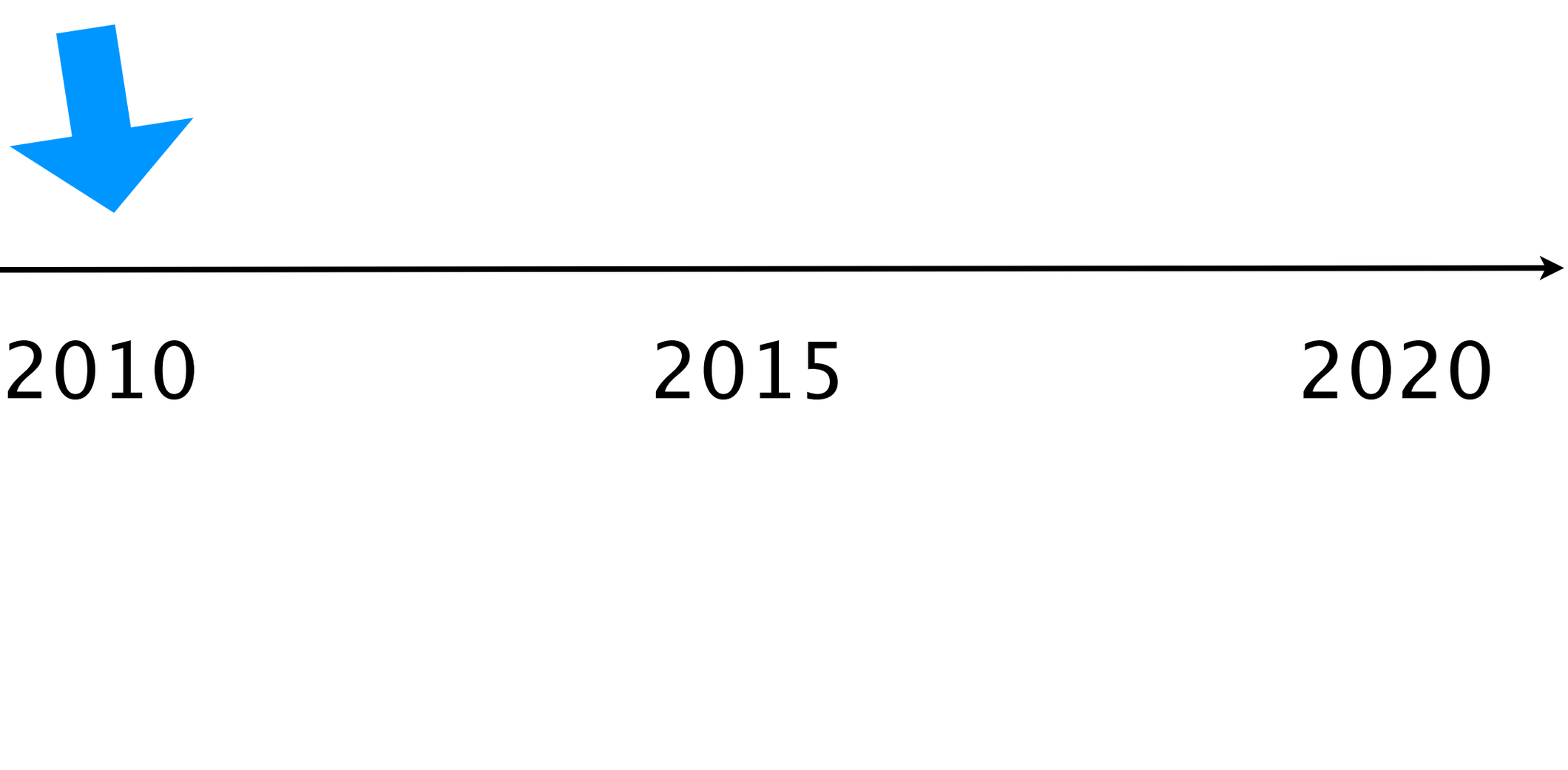
- プロセッサの向上: 周波数 → コア数
- 並列に書くしかない
- 速さ → 人類の GNH 向上

近未来 LL

近未来 Timeline

現状認識

いまここ



気にすること

- 並列に動くか
  - ≠ 並行
  - Native thread で走るか
- どんなモデル、パラダイムを実装しているか

- Native thread
  - 5.6 から ithread
    - thread ごとに VM
    - default で shared nothing
- 6.0 から STMなどたくさん
- ライブラリ
  - POE, AnyEvent, Coro, ...

- Native & green thread (実装による)
  - ruby1.8: green
  - ruby1.9: native, GIL あり
  - JRuby, MacRuby: native, GIL なし \o/
- ライブラリ
  - Revactor, EventMachine, NeverBlock, ...

- Thread
  - CPython: GVL あり
  - Jython, IronPython : なさげ
    - Jython: system 216% cpu 12.285 total

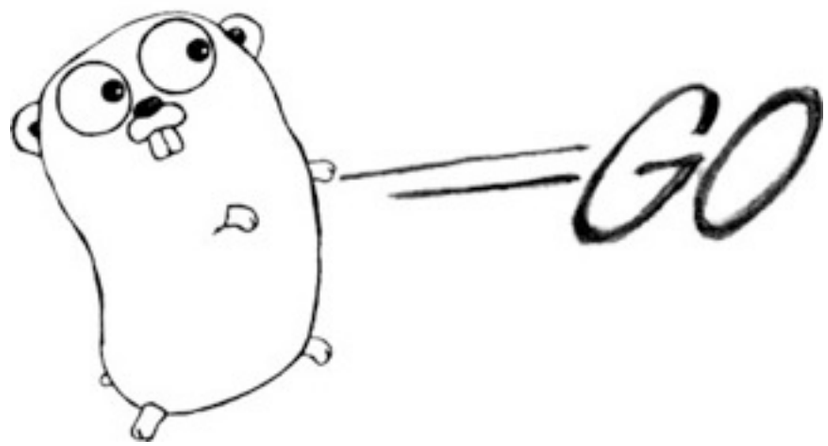
- no threads, fork it

Haskell

- 最終兵器 ~~彼女~~ Haskell
- なんかいろいろあってすごい
  - Parallel Haskell
  - Concurrent Haskell
  - STM
- Native thread?

- Actor モデルのメッセージパッシング
- Native thread
  - beam.smp 193.2%
- マシンの中/外が透過的

- いいとこどり
- JVM ベース
  - Native thread
- Actor モデルのメッセージパッシング



- goroutine
  - Native thread
    - runtime.GOMAXPROCS(NCPU)
- Channel
  - メッセージパッシング

Javascript

- HTML5 で WebWorkers が！
  - メッセージパッシング
  - Native thread っぽい?
    - WebKit のソースによると
- 識者に聞く

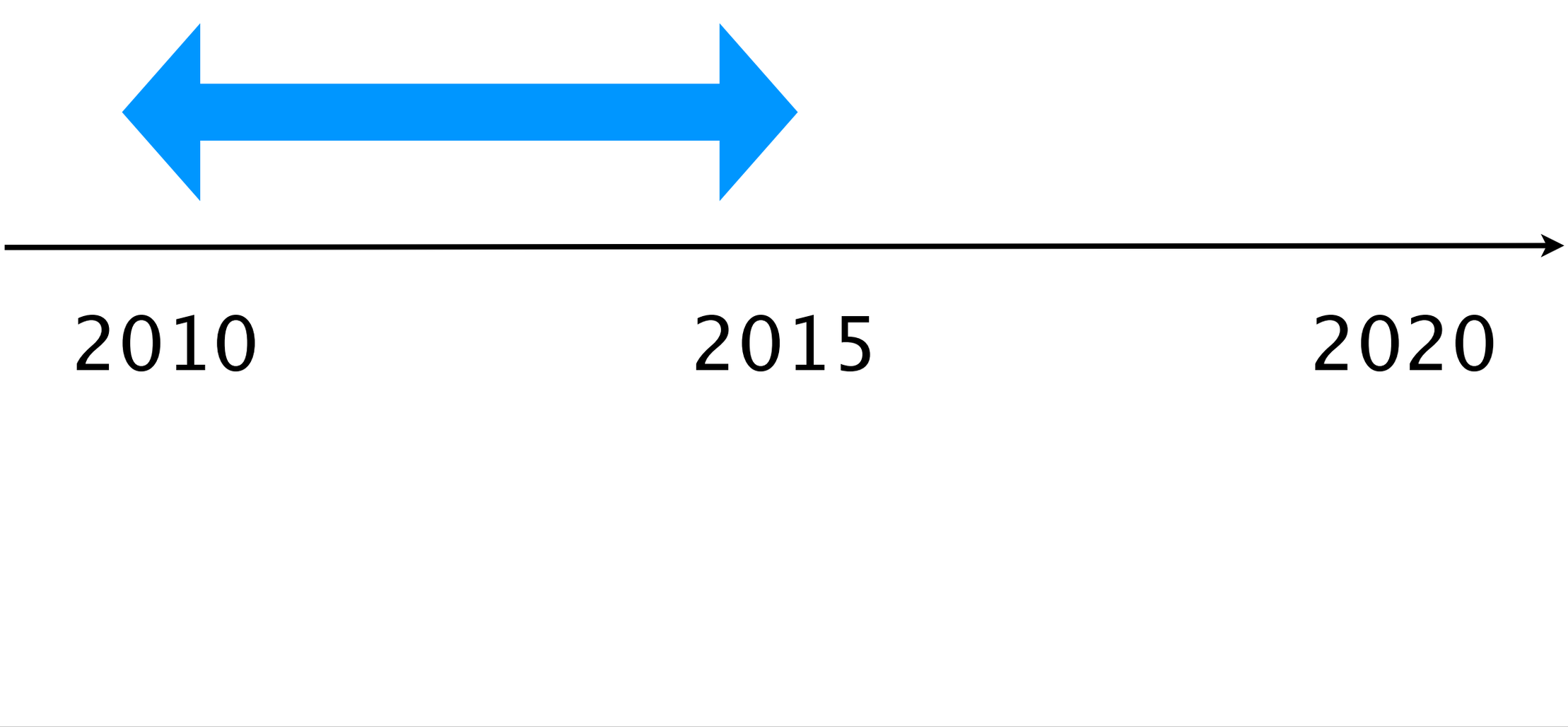
現状まとめ

	Perl	Ruby	Python	PHP	Haskell	Erlang	Scala	go	Javascript
Native thread	◎	△	△	×	○?	◎	◎	◎	○?
Model, Paradigm	Thread, STM	Thread, Fiber	Thread	Process	STM すごい何か	Message Passing	Thread, Message Passing	Message Passing	Message Passing
特記事項	shared-nothing by default				研究熱心	マルチコアでもクラスタでも	いいとこどり	goroutine	Web Workers

2010–2015

Timeline

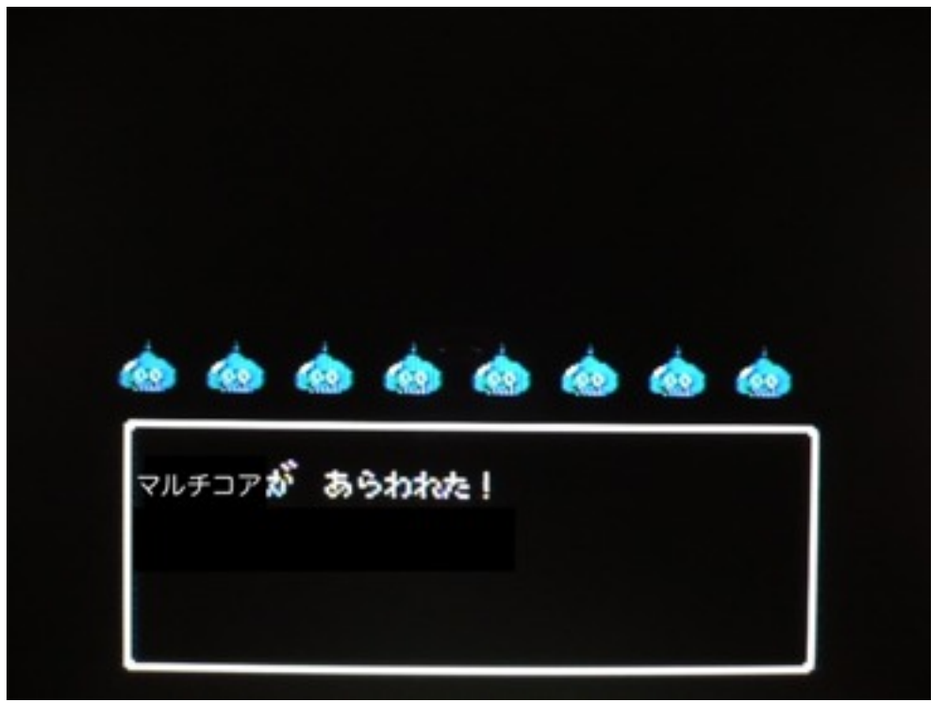
このへん




GRANITE MOUNTAIN
POWER
WALL
WHAT
MYSTIC
POWER
DOES
THIS
WALL
HAVE?
TRY
THIS...
STAND WITH YOUR RIGHT
FOOT SIDEWAYS AGAINST
THIS WALL AND PUT YOUR RIGHT
CHEEK AGAINST THE WALL, THEN
TRY TO LIFT YOUR LEFT FOOT.
GO AHEAD
AND TRY IT!

電力がやばい

↓


マルチコアが あらわれた！

JVM ベース処理系の時代

- Native thread, マルチスレッドの性能
- ポータブル
- 例:
  - JRuby, Jython, Scala, Clojure, ...

でも待って

http://www.flickr.com/photos/johnkay/3331613033/

でも待って

- マルチスレッドプログラミングは死ぬ
  - デッドロック, レース, 再現性のないバグ, ...
- 天才にとってさえ難しい
- スレッドはプリミティブすぎる
  - アセンブリで書くようなもの

抽象が必要

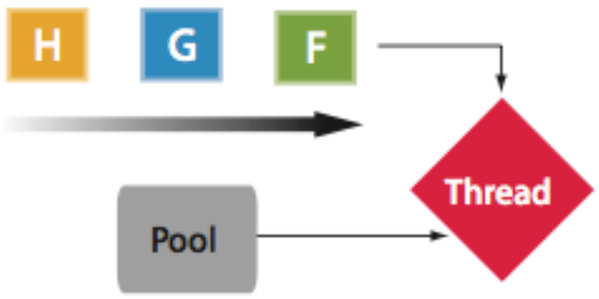
- 非同期 I/O
- タスクキュー
- データフロー
- メッセージパッシング
  - イベント駆動

非同期 I/O

- C10k 問題
- Event
  - AnyEvent (Perl)
  - EventMachine (Ruby)

タスクキュー

- 並列に実行したいタスクを
  どんどんキューにつっこむ
  - :) 分かりやすい
  - :(依存関係があると...?
- 例:
  - Grand Central Dispatch → Mac
  - Java 7 → JVM ベースの処理系
  - OpenCL



http://images.apple.com/macosx/technology/docs/GrandCentral_TB_brief_20090903.pdf

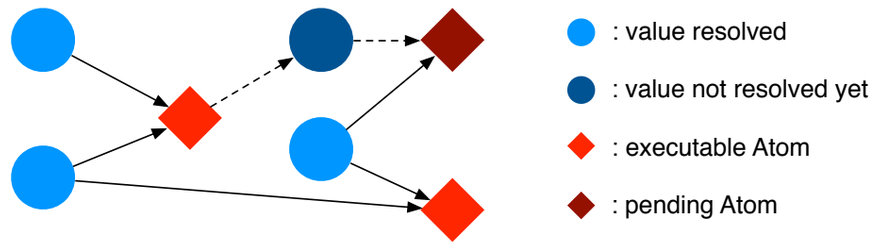
http://en.wikipedia.org/wiki/File:Gcd_icon20090608.jpg

http://www.khronos.org/opencl/

データフロー

- 依存関係をつなぐ
- 必要なデータがそろったら、自動的に次の処理に進む

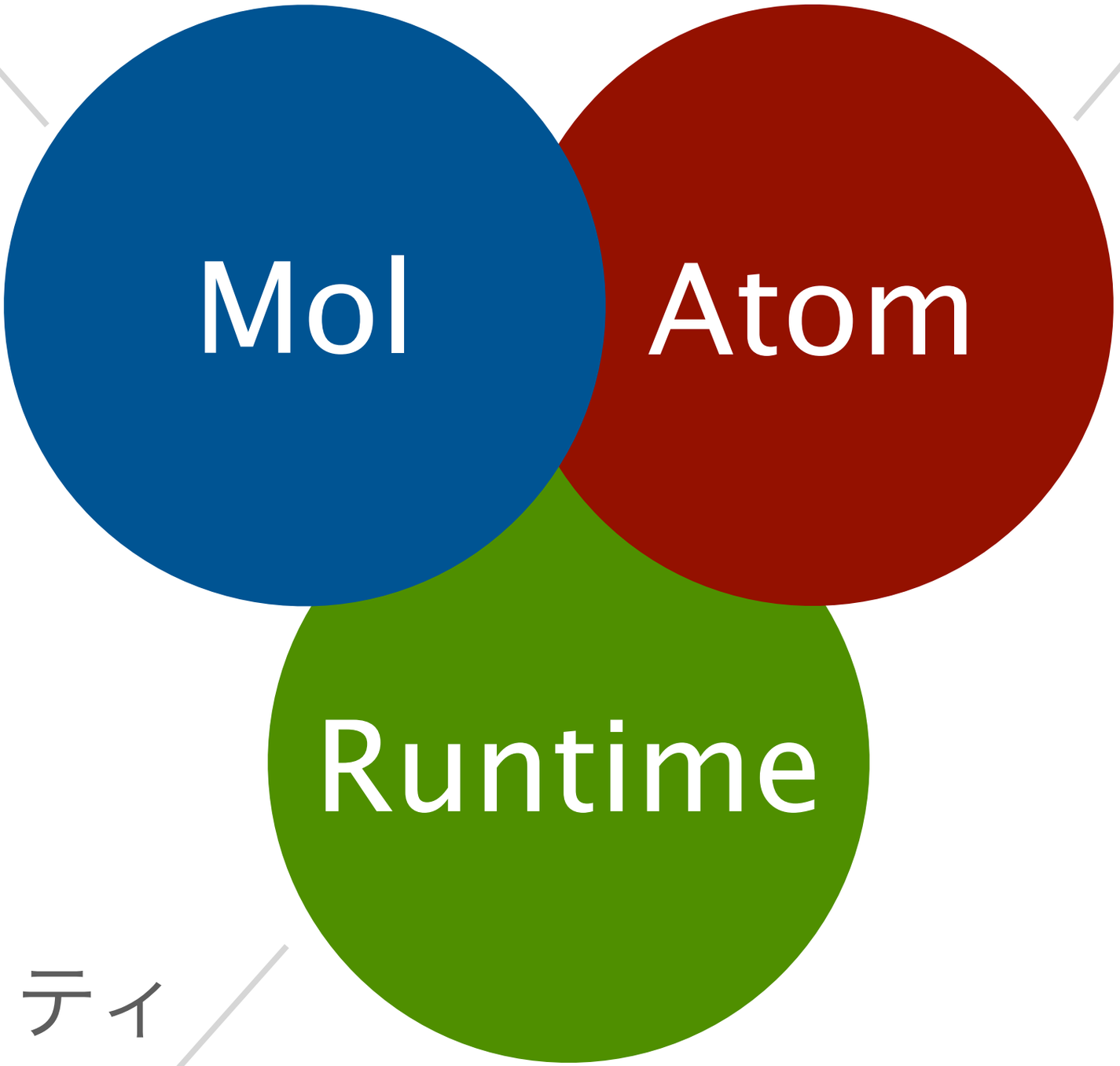
- Out of Order
- Future パターン
  - 投げっぱなしジャーマン
  - 別スレッドに計算させといて、自分はそのまま実行を継続
  - 別スレッドが終わったら通知をもらう



Molatomium

並列 DSL

直列 C



ポータビリティ

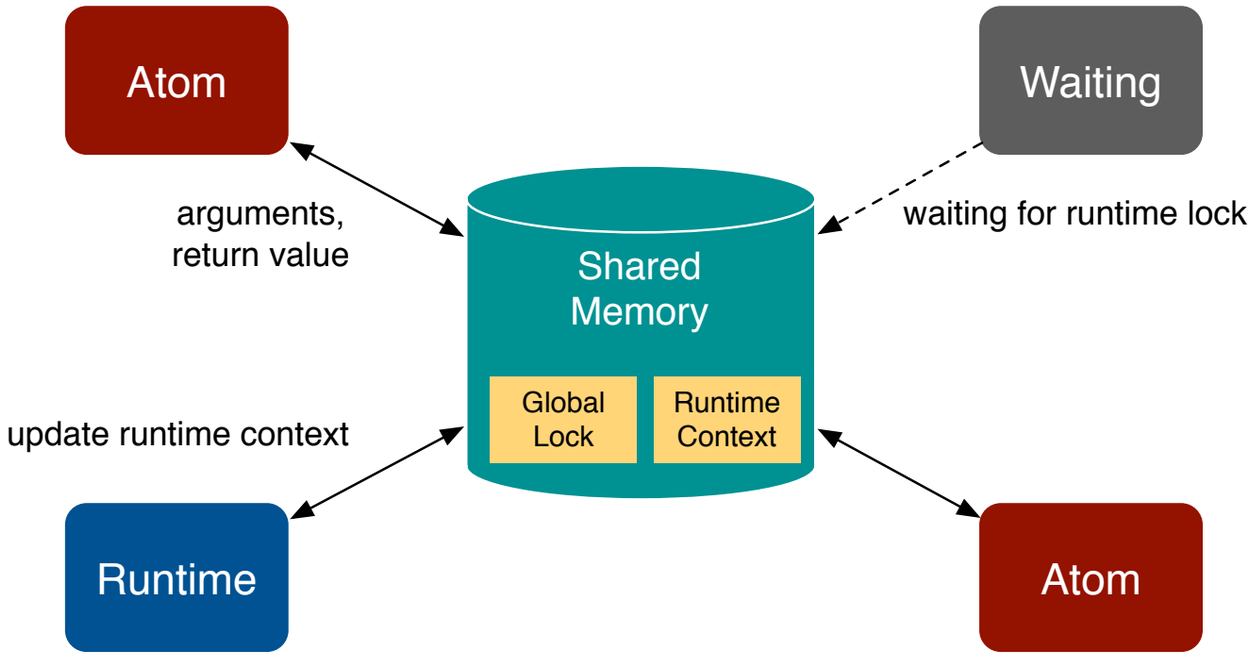
Mol

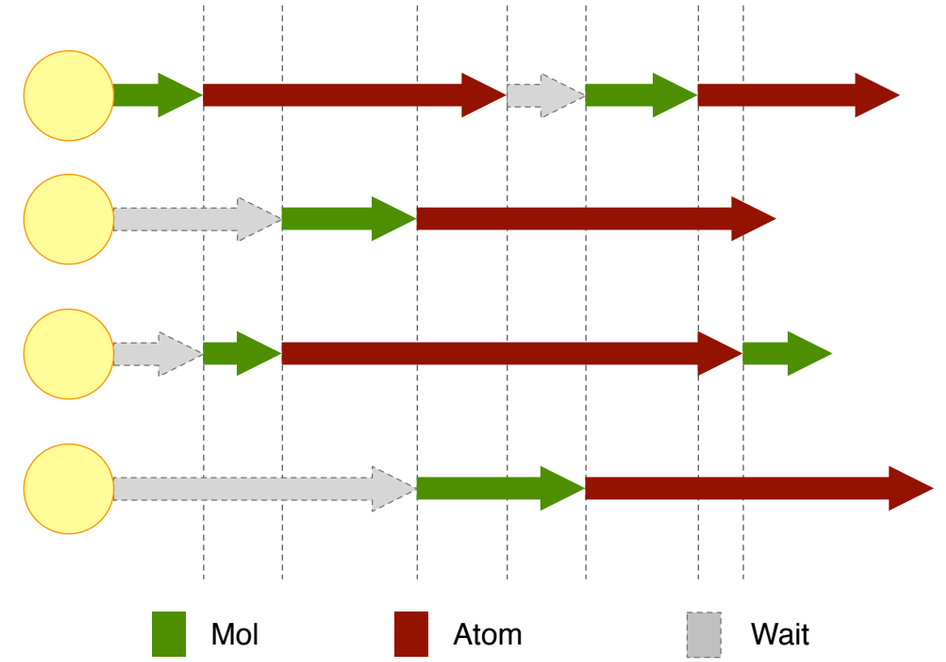
- 宣言的、データフロー
  - いちいち同期書かない
  - Atom (直列コード) を遅延並列実行、メモ化
- C みたいな Haskell みたいな 並列 DSL

```
main() {
  frame(frame_no, input) {
     local loop;
     local edge[-1..9] outside(0);
     local pocs[-1..9] outside(0);
     local ccr [-1..9] outside(0);

     loop     = loop_count();
     edge[i] := edge_atom(input, i);
     ccr [i] := ccr_atom(edge[i-1], edge[i], edge[i+1], i);
     pocs[i] := i % 2 == 0
       ? pocs_even_atom(ccr [i-1], ccr[i], ccr [i+1], loop, i)
       : pocs_odd_atom (pocs[i-1], ccr[i], pocs[i+1], loop, i);
     return &pocs;
  }

  sync for (i in [0..30])
     put_frame_atom(frame(i, get_frame_atom()));
}
```

Runtime

- 各コアに VM がいる
- ここがキモ

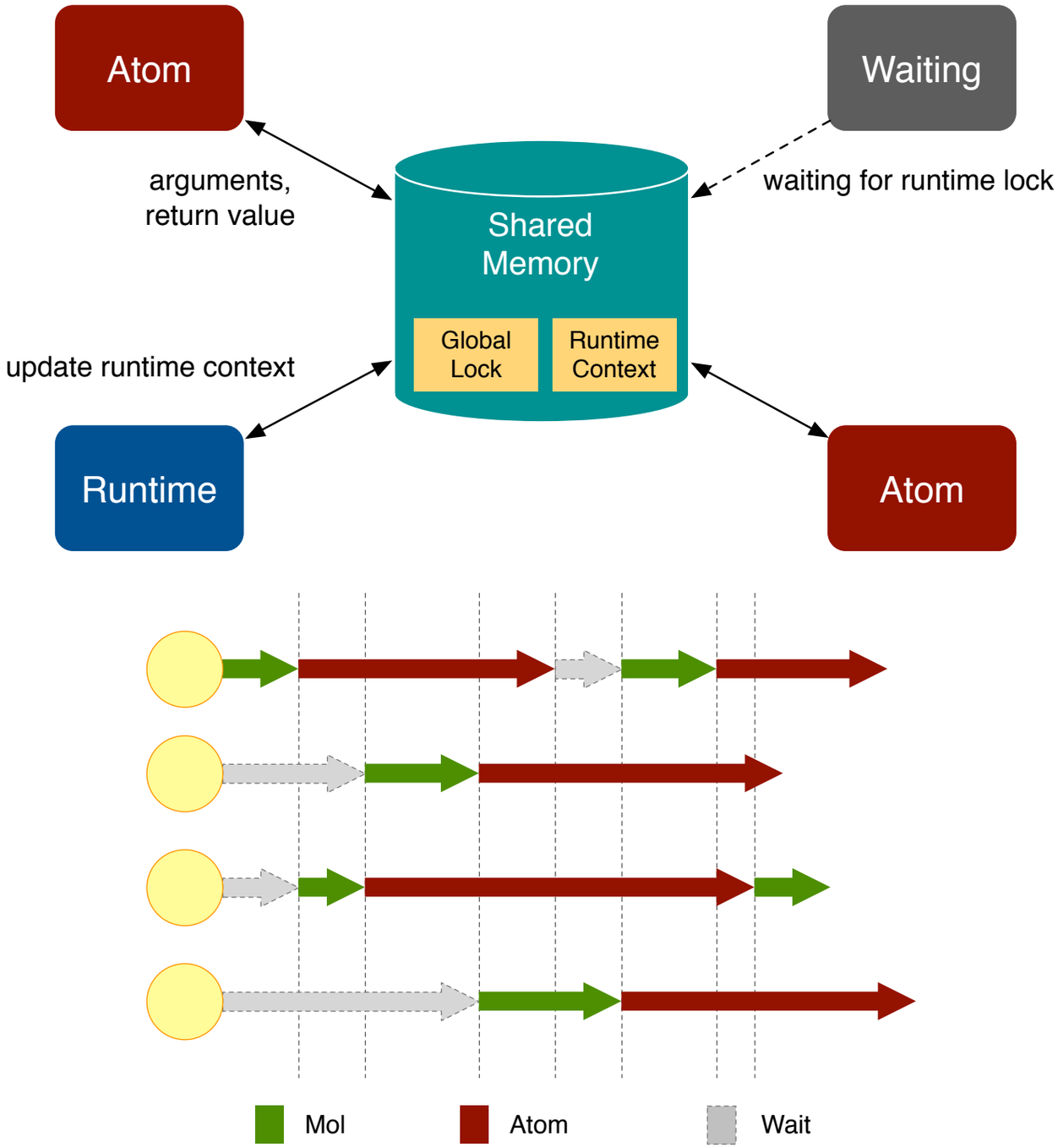




Runtime

- 各コアに VM がいる
- ここがキモ

続きは論文で

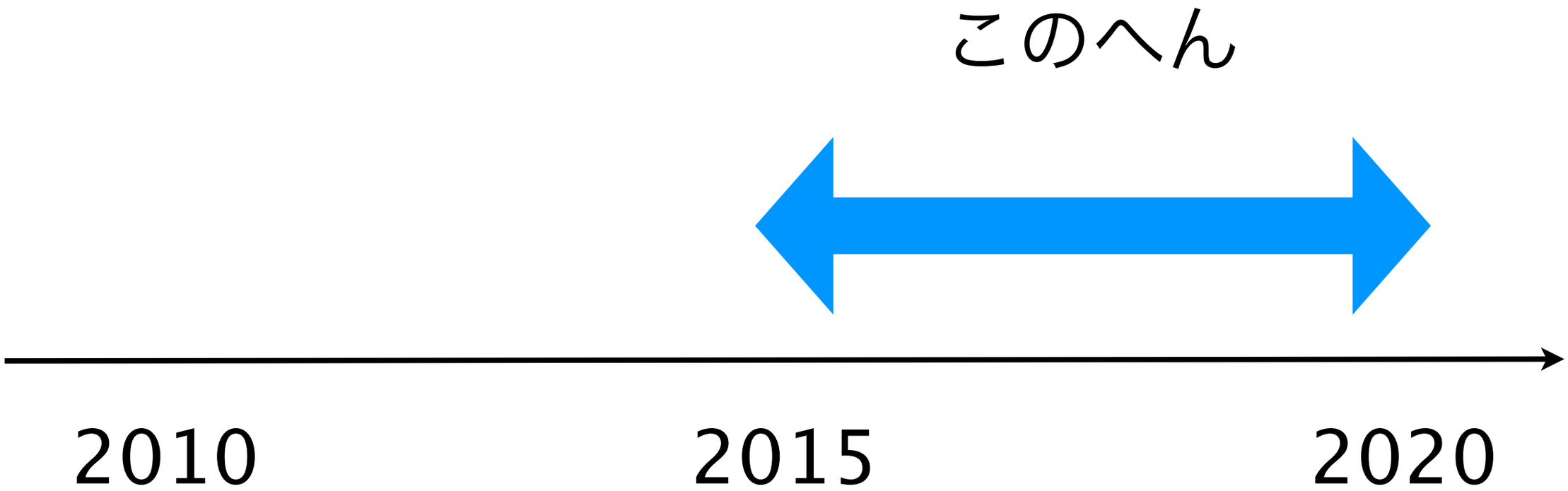


2010–2015 まとめ

- マルチコア期
  - JVM ベースの処理系が流行りそう
- 生でスレッド使うのは無理ゲー
  - 抽象を使おう (非同期 I/O, タスクキュー、データフロー、メッセージパッシング)

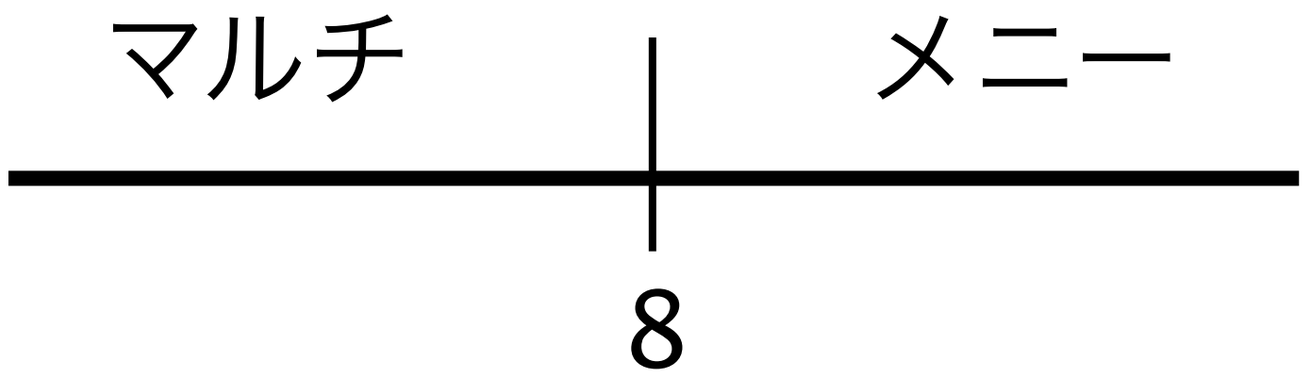
2015–2020

Timeline



メニーコア期

マルチ | メニー



FAQ

- そんなたくさんあって何に使うの?

FAQ

- そんなたくさんあって何に使うの?

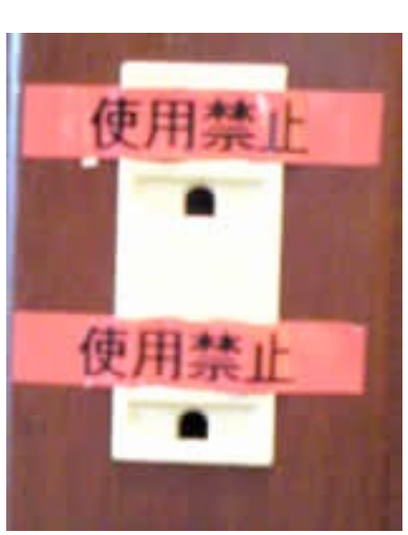
電源難民問題を解決

http://www.speedia.co.jp/~namisato/2008_05.html

FAQ

- そんなたくさんあって何に使うの?

電源難民問題を解決



http://www.speedia.co.jp/~namisato/2008_05.html

P = f V^2

1MHz が 1000 コア → 1GHz 並の性能が!

Memory Wall

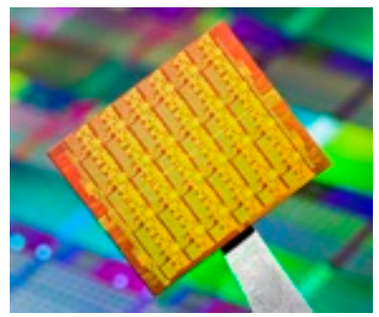
メモリがやばい

Memory Wall

- 容量が増えない
- 電気を食う
- 帯域が足りない
  - 100コアを超えるとダメげ
    - by “Can Manycores Support the Memory Requirements of Scientific Applications?” / A4MMC 2010

Memory Wall

- 共有メモリ → 分散メモリ
- 例: Intel SCC
  - 共有メモリ遅い
  - メッセージパッシング の HW がある



http://techresearch.intel.com/articles/Tera-Scale/1826.htm

メッセージパッシング

- Actor モデル
  - Erlang, Scala, go, Javascript (HTML5, WebWorkers)
  - 他の LL でもライブラリがある

いきなり...?

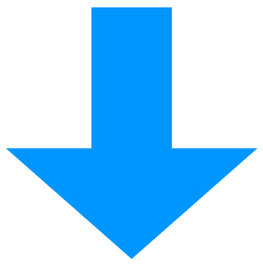
- いきなりのパラダイムシフトは
  きついかも
- 処理系はメッセージパッシング、
  ユーザは慣れたパラダイム
  - スレッド、イベント、タスクキュー

幸せな未来

たくさんのコア

↓

ひとつのコアに見える



並列化してないコードが速く!

http://dragon-quest.jp/five/monster/11kingusuraimu.html

幸せな未来

たくさんのコア

↓

ひとつのコアに見える

並列化してないコードが速く!

http://dragon-quest.jp/five/monster/11kingusuraimu.html

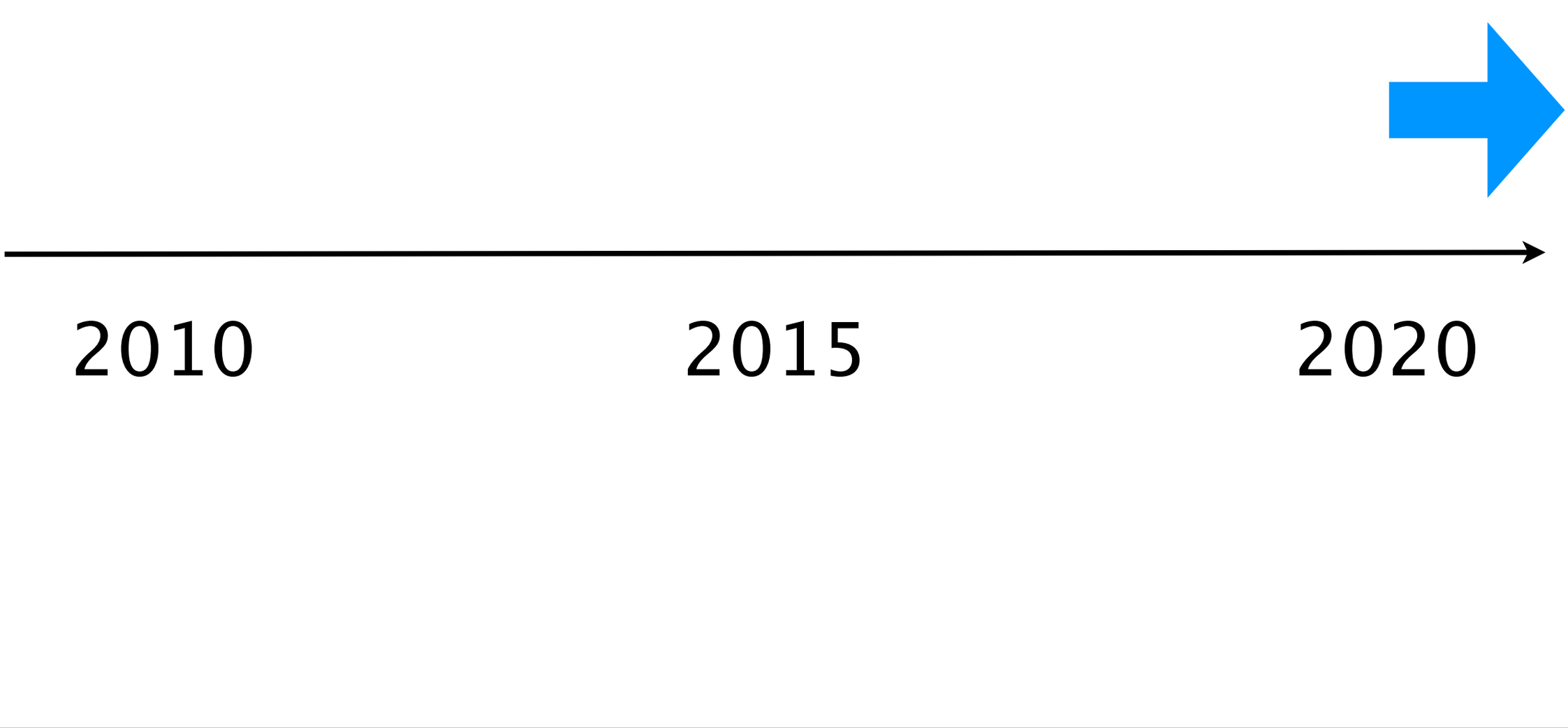
2015-2020 まとめ

- メニーコア期
- メモリがやばい → 分散メモリに
- メッセージパッシングが主流になりげ

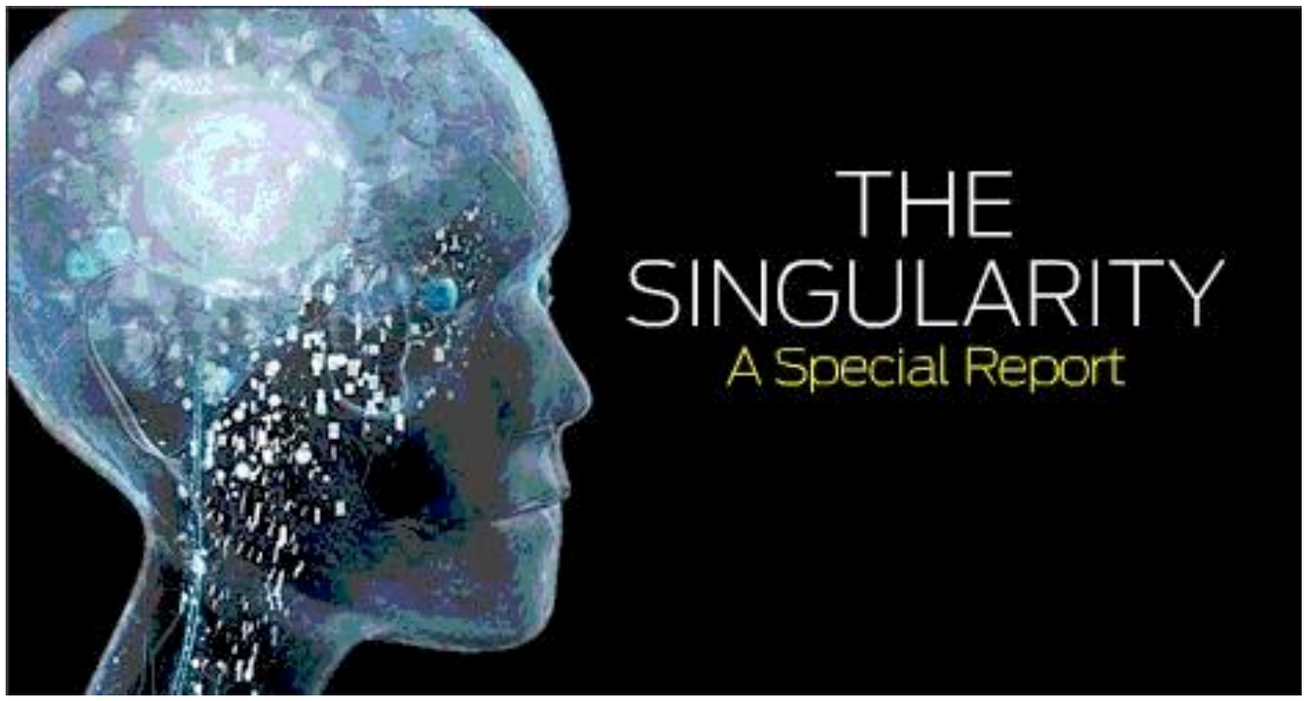
2020 ->

Timeline

このあと

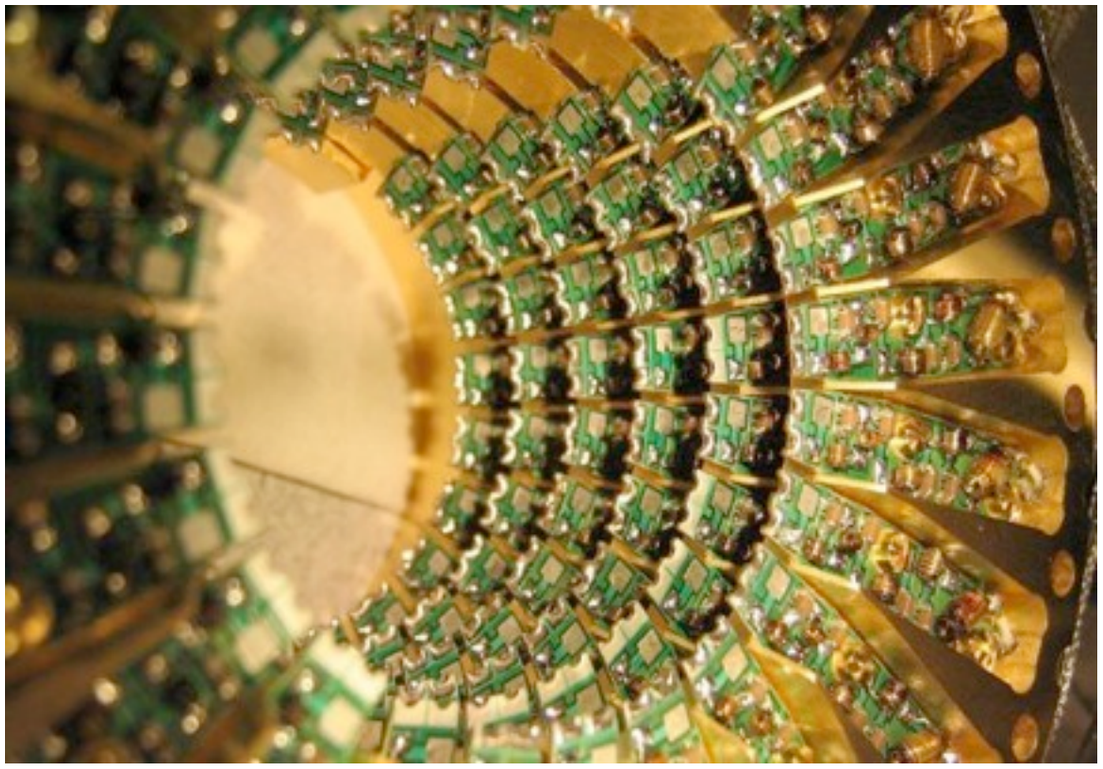


正直未来すぎて想像つかない



http://karlnordstrom.ca/ideas/?p=6

量子コンピュータ

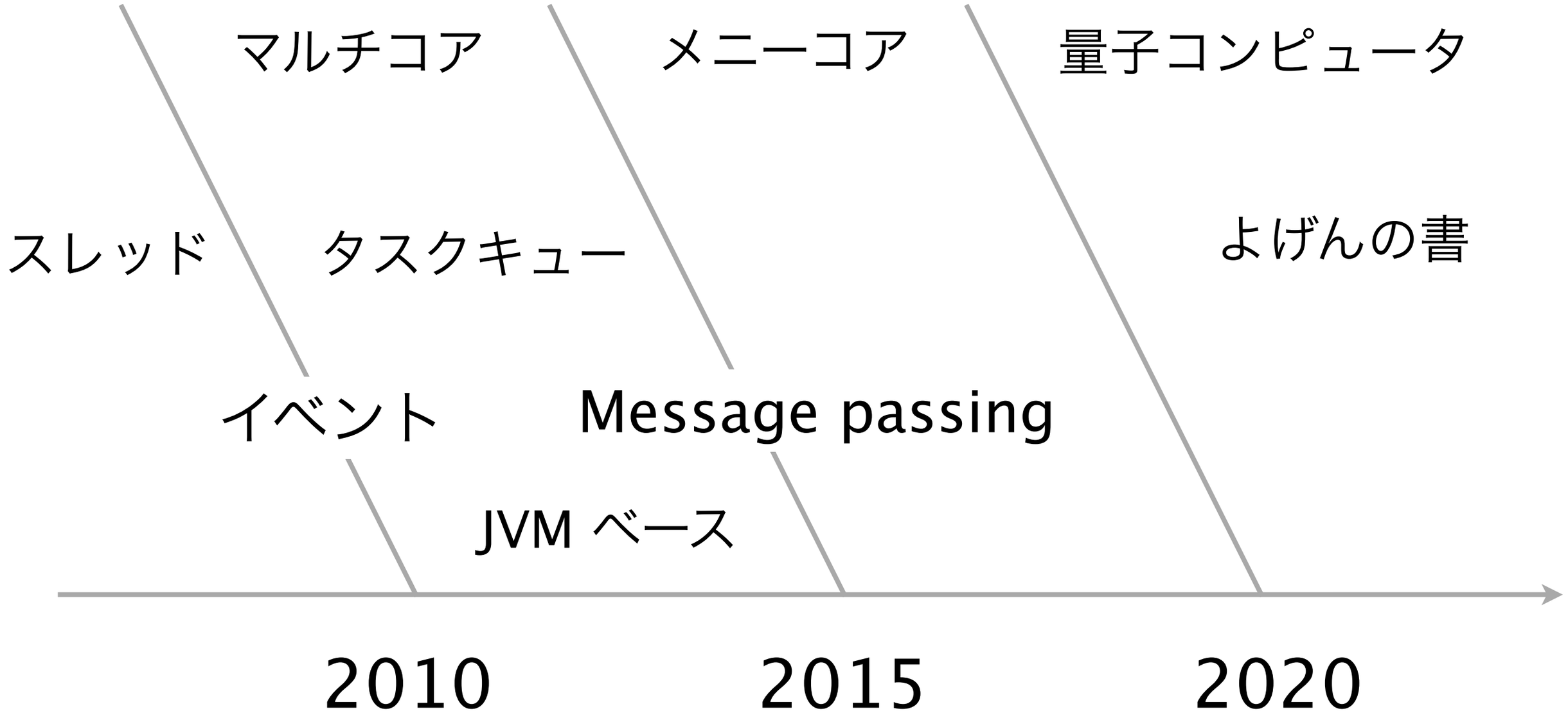


http://www.fastcursor.com/computers/quantum-computer-photo-gallery.asp

ここまでの

まとめ

近未来 Timeline



未来の並列 LL

を妄想

理想の並列 LL

P

- どんな構成でも OK (Portable)
  - マルチコア、メニーコア、ヘテロ
- コア数に応じてスケールする (Performance)
- ラク (Productivity)
  - 並列を意識せず書いて OK

現実的に

なろう

現実的な並列 LL

- ~~どんな構成でも OK (Portable)~~
  - ~~マルチコア、メニーコア、ヘテロ~~
- コア数に応じてスケールする (Performance)
- ラク (Productivity)
  - ~~並列を意識せず書いて OK~~

LL って?

- プログラマの負担が Light
- 現実、現場的
  - 必要に迫られてつくられるもの
  - アプリ次第、 DSL

方向性: 並列 DSL

- 直列コードをつなぐ並列 LL
  - 直列コード: C, Ruby, Perl, PHP, ...
  - Communicating Sequential Process
- LL 向きの並列性
  - 粒度おおきめ
- Glue としての LL, メタ LL

使いやすさって?

- 人が理解できること
  - 複雑なモデルは使われない
  - 見えないものを想像するのはむずい (帯域など)
- 人は複数のことを同時に考えられない
  - マルチタスクは生産性を 40% 下げる

(by Harvard Business Review
http://blogs.hbr.org/bregman/2010/05/how-and-why-to-stop-multitaski.html)

未解決問題

- 局所性
- 帯域
- タスクとデータ転送のオーバーラップ
- タスクの粒度調整
- ヘテロ

http://dragon-quest.jp/five/

今がチャンス

- アーキテクチャ激変のまっ最中
- デファクトをとれるかも
- ついに C が...

制約

- 本当の制約は何か
  - 物理的なもの (チップ面積、消費電力)
- しがらみとか実装上の制約は人工的

Re-Discover

- 昔、見切りをつけた技術が今こそ使えたりする

例:

- 例:
  - GC
  - Javascript
  - メッセージパッシング
  - データフロー

ここまでまとめ

- 現実的近未来並列 LL
  - 生産性と性能は欲しい
- いまがチャンス
- アプリ次第

Application

Application

- アプリが言語、モデルをドライブする
- 言語、モデルがハードウェアをドライブする

どんな?

1. 時間がかかりすぎて
あきらめていたこと

例: Ray Tracing

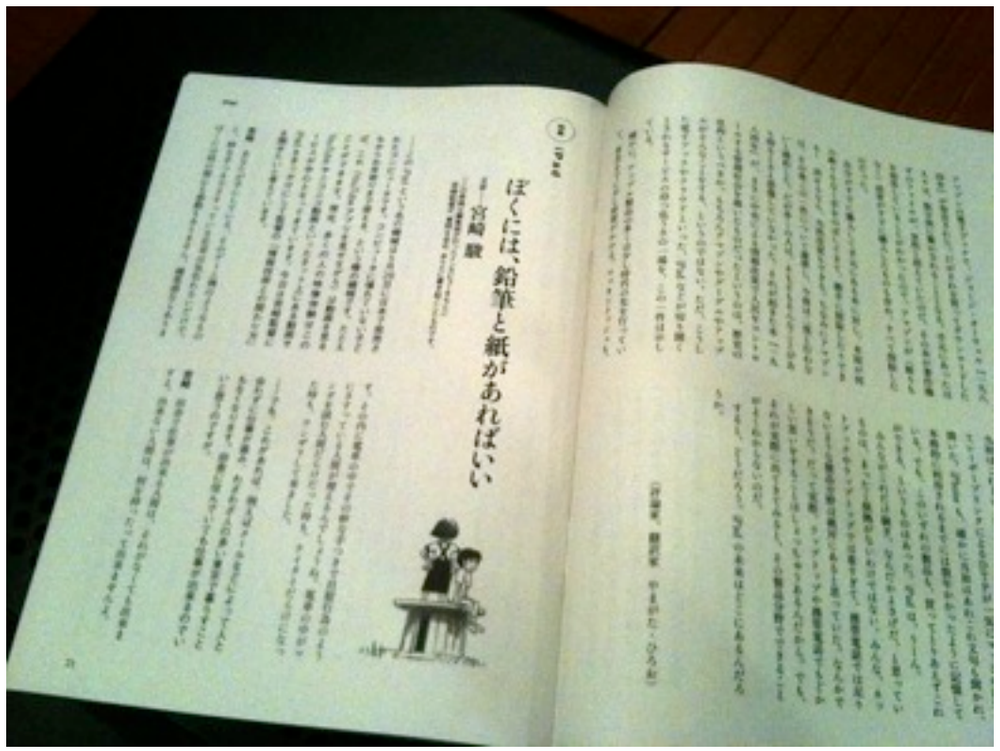
http://lucille.sourceforge.net/gallery/0_1_3/index.html

スピードが質に転換する

- 速くなるだけで世界が変わることがある
- 例:
  - 手術シミュレーション
  - 会話認識
    - 一日かかっていた処理がリアルタイムに

2. つくる系

“あなたは消費者になってはいけない。
生産する者になりなさい。”



http://blogs.itmedia.co.jp/yasusasaki/2010/07/ipad-ab70.html

2. つくる系

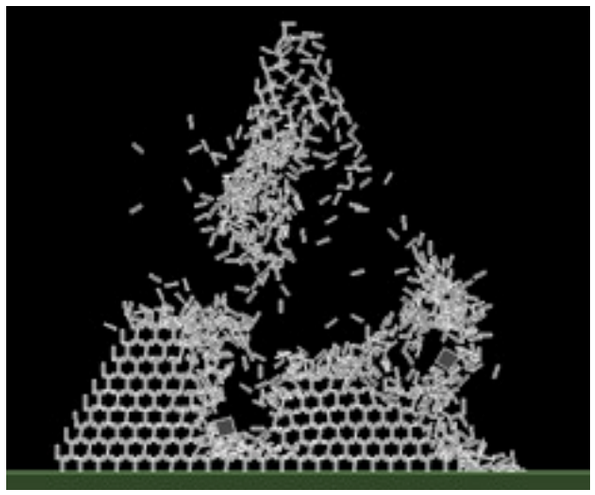
- try-error のサイクルが

  劇的に短縮されると...?

- 例
  - Pixelmator
  - 動画編集

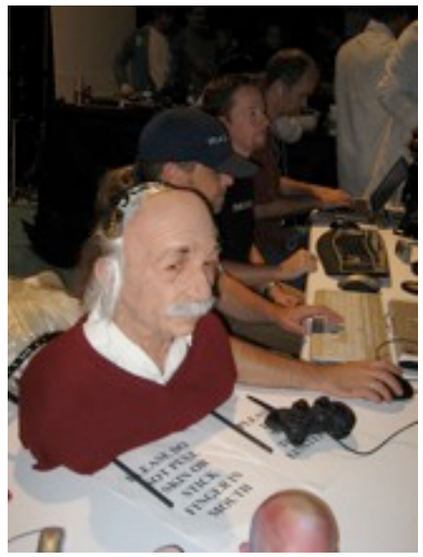
3. 合わせ技

計算量ドカ食いアプリをマッシュアップ



http://commons.wikimedia.org/wiki/File:Box2d-screenshot.png

http://www.flickr.com/photos/overig/4639331740/



http://www.flickr.com/photos/nh567/1406671607/

4. 拡張系

- プラグイン、ボット、拡張といわれるもの
- 並列で動くものが対象
- AI、脳みそ
- ロボット

その他

- 3次元映像を編集
- 2次元 → 3次元

以上まとめ

近未来並列 LL

- 並列プログラミングで快適な世界をつくろう
- 並列 LL
  - 生産性と性能を両立や！
- 今すぐできること
  - メッセージパッシングに慣れとくとよさげ
- アプリケーション
  - 速くなったら世界が変わるものって?