OpenCL™と最近のGPUトレンド

日本AMD株式会社
ジャパンエンジニアリングラボ（JEL）
西　剛伺

目次

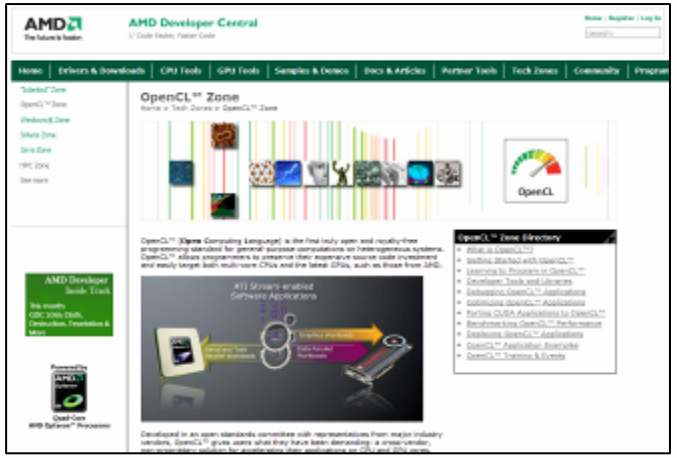
- OpenCL™について
- 最近のGPUトレンド - AMD/ATI GPUを例に -

OpenCL™について

OpenCL™: 革新的な開発環境
GPGPU機能の幅広い適用を可能に

- **業界標準のAPI:** ヘテロジーニアスアーキテクチャのためのオープンな開発プラットフォーム。マルチプラットフォームに対応。
- CPUとGPUの双方を活用し、バランスの取れたシステムを構築可能
- **業界の幅広いサポート:** AMD, Apple, IBM, Intel, Nvidia, Sony等のアーキテクトによって策定
- **急ピッチでの開発:** 2008年12月に承認され、各ベンダが実装
- **モーメンタム:** メインストリームの開発者が大きな関心を寄せている

より多くのGPGPU対応
アプリケーションが世に出てくる



AMDでは開発者向けサイト(developer.amd.com)で
OpenCL™に関するさまざまな情報を発信しています。

OpenCL™ Zone:
http://developer.amd.com/openclzone

OpenCL™の概要

Language Specification

- C-based cross-platform programming interface
- Subset of ISO C99 with language extensions - familiar to developers
- Well-defined numerical accuracy - IEEE 754 rounding behavior with defined maximum error
- Online or offline compilation and build of compute kernel executables
- Includes a rich set of built-in functions

Platform Layer API

- A hardware abstraction layer over diverse computational resources
- Query, select and initialize compute devices
- Create compute contexts and work-queues

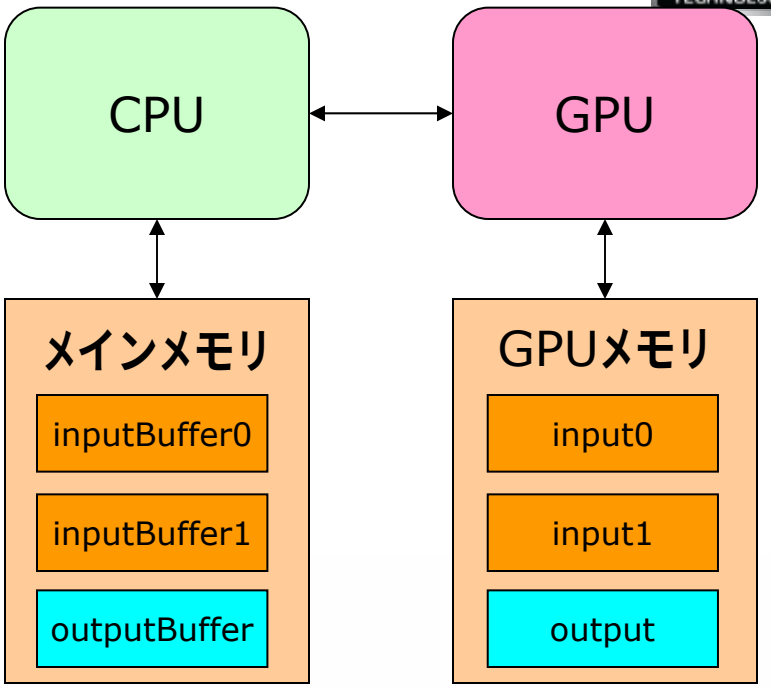
Runtime API

- Execute compute kernels
- Manage scheduling, compute, and memory resources

OpenCL™プログラム例

```
int main()
{
    …
    // ICD based init
    ICDinit(&platform);

    // Setup
    setupCL(CL_DEVICE_TYPE_GPU, …);

    // Compile, Load Kernel
    compileLoadCL(…, "vec_mult.cl");

    // Create and Initialize buffers
    input0 = clCreateBuffer(context, …);
    input1 = clCreateBuffer(context, …);
    output = clCreateBuffer(context, …);

    // Execute kernel
    runCLKernel(…, &input0, &input1, &output, …);

    // Wait for kernel completion
    status = clFinish(cmd_queue);

    // Read data from CL device to CPU memory
    clEnqueueReadBuffer(cmd_queue, …);

    // Wait for read completion
    status = clFinish(cmd_queue);
    …
}
```

main.cpp (ホストコード)




```
__kernel void vec_mult (__global const float *a,
                        __global const float *b,
                        __global float *c)
{
    int gid = get_global_id(0);
    c[gid] = a[gid] * b[gid];
}
```

vec_mult.cl (カーネルコード)

AMD The future is fusion

OpenCLカーネルの文法基礎 – Cとの違い

```
__kernel void vec_mult (__global const float *a,
                        __global const float *b,
                        __global float *c)
{
   int gid = get_global_id(0);
   c[gid] = a[gid] * b[gid];
}
```

1. **カーネル関数の定義には、**__kernel**もしくは**kernel**修飾子をつける。**
2. **カーネル関数のポインタ引数は、**__global, __constant, __local**修飾子のいずれかをつけて宣言する。**
3. ID**を取得してアドレッシング。**

＜注意＞
“__”はアンダースコア2つ分です。

詳しくは
The OpenCL Specification, *Version: 1.1***に記述があります。**

OpenCL™のプログラミングモデル

- **実行モデル**

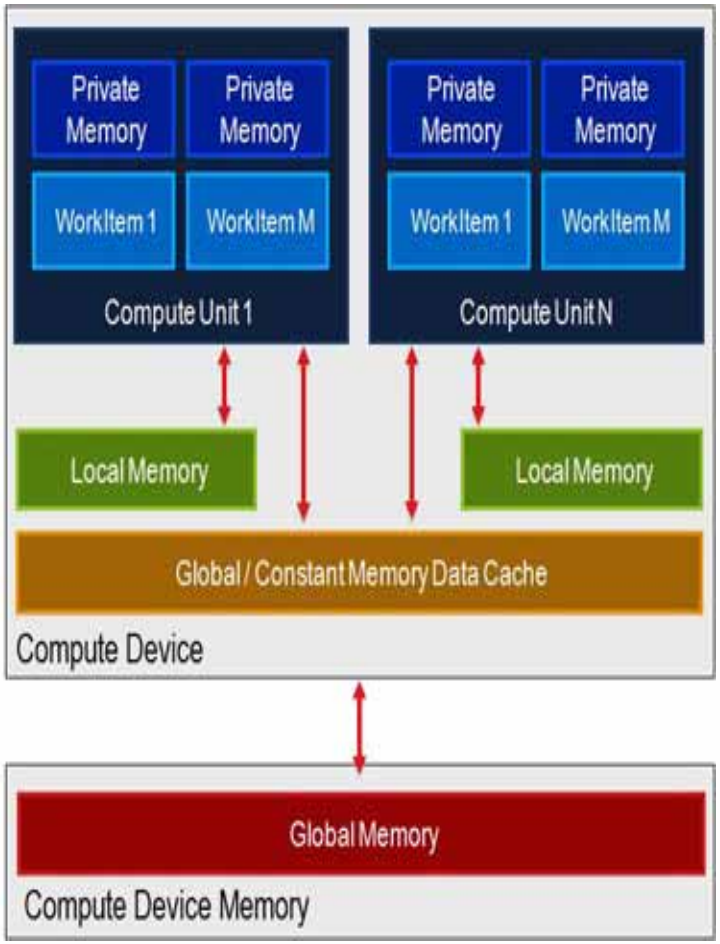
  ***コンピュートカーネル*が基本的な実行ユニット**

  in-order**もしくは**out-of-order**実行が可能**

  カーネルは*データ並列* (GPU) **もしくは**
  タスク並列 (CPU)

  カーネルのためのN**次元実行ドメイン**

  同期、コミュニケーション用に*work-item***を**
  ***work-group*にグループ分け可能**



- **メモリモデル**

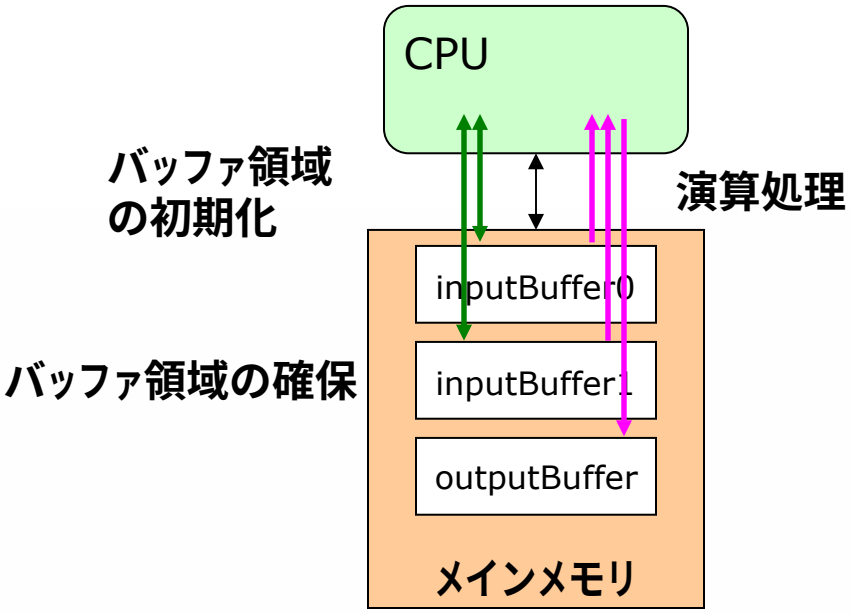
  マルチレベルのメモリモデル:
  private, *local*, *constant*, *global*

詳しくは
The OpenCL Specification, *Version: 1.1***に記述があります。**

既存のPCアプリケーションプログラムの開発環境

CPUではOSが動作
- Microsoft® Windows®
- Linux®

プログラミングには
- Microsoft® Visual Studio®
- gccなど



GPGPU対応アプリケーションプログラムの開発環境

CPUではOSが動作
- Microsoft® Windows®
- Linux®

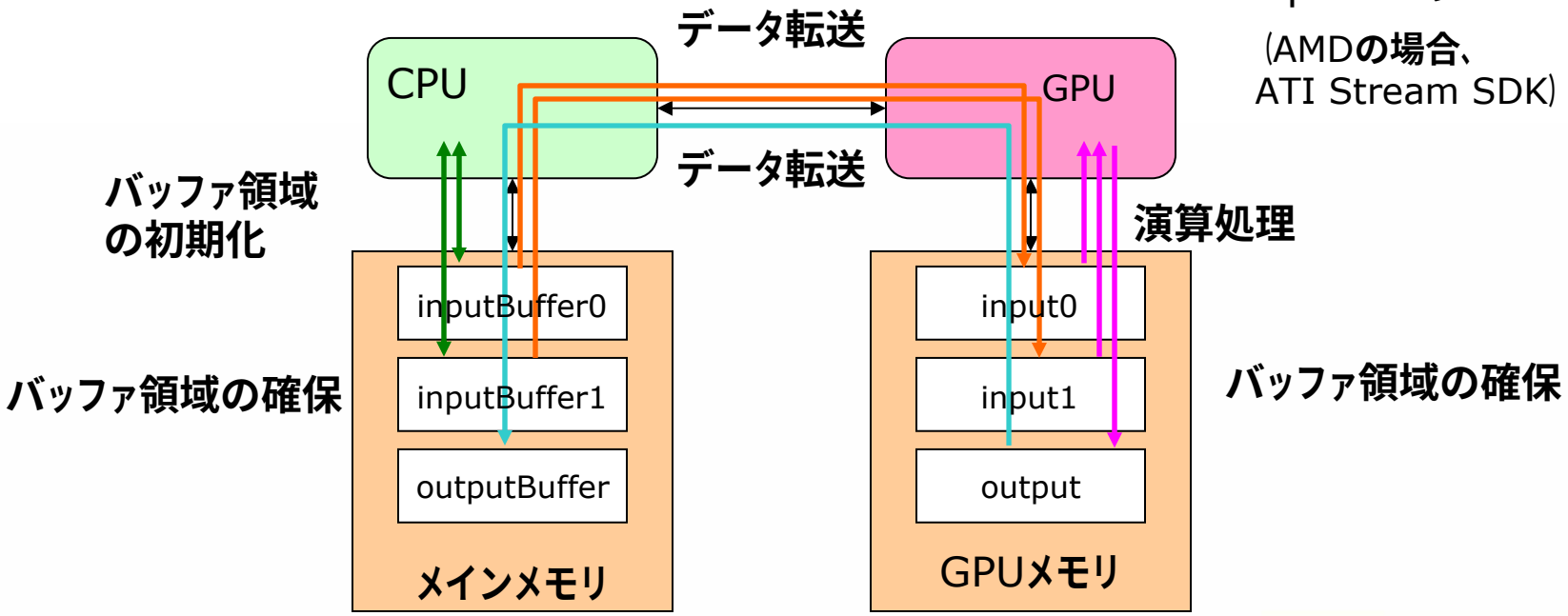
プログラミングには
- Microsoft® Visual Studio®
- gccなど

CPUとGPU間のコミュニケーションは？
- GPUドライバ
- GPGPU専用のAPI (CPUからコール)

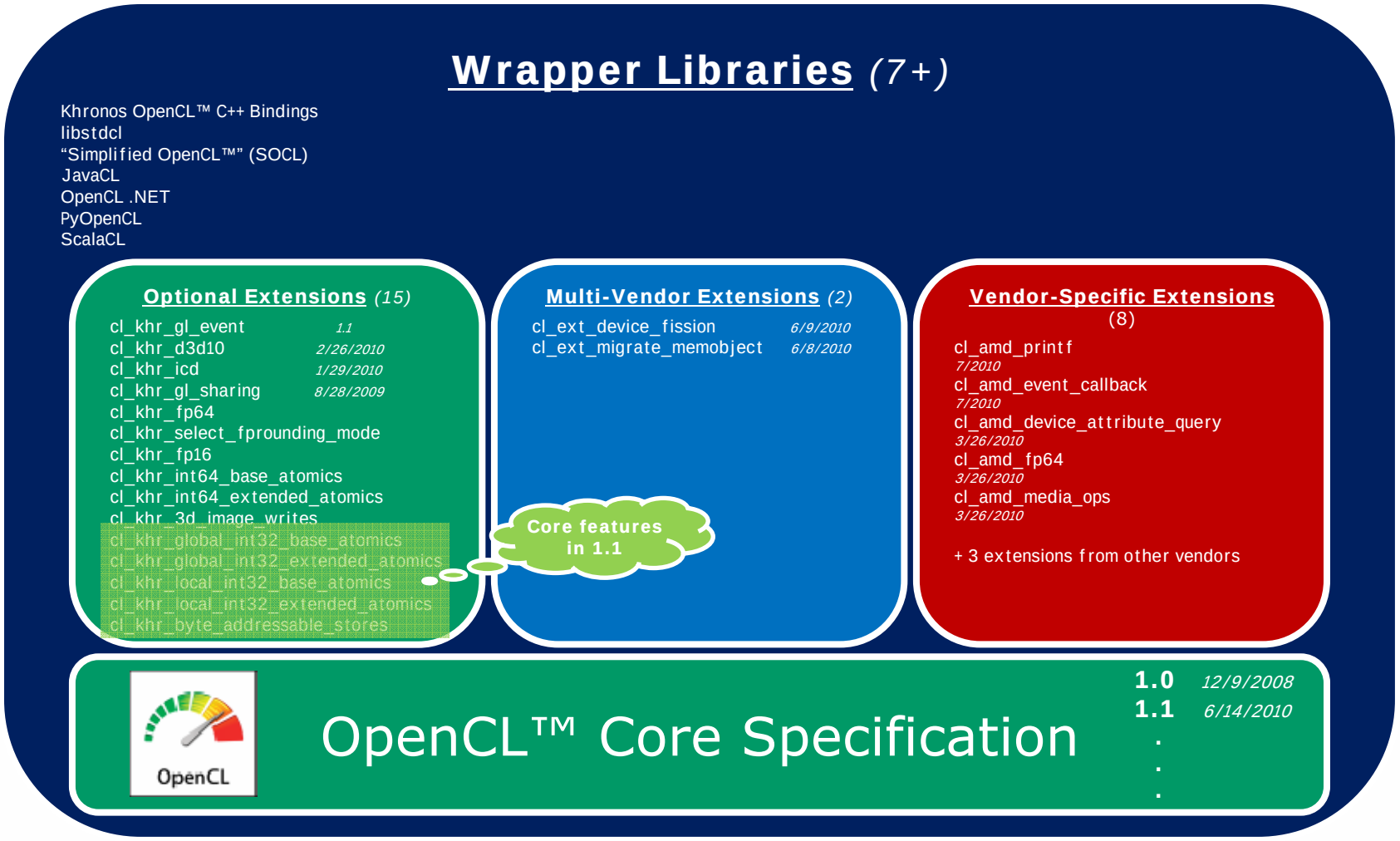
GPUにはOSはない

GPUコード (カーネル)
プログラミングには
- GPU専用のコンパイラ

OpenCLのSDK
(AMDの場合、ATI Stream SDK)

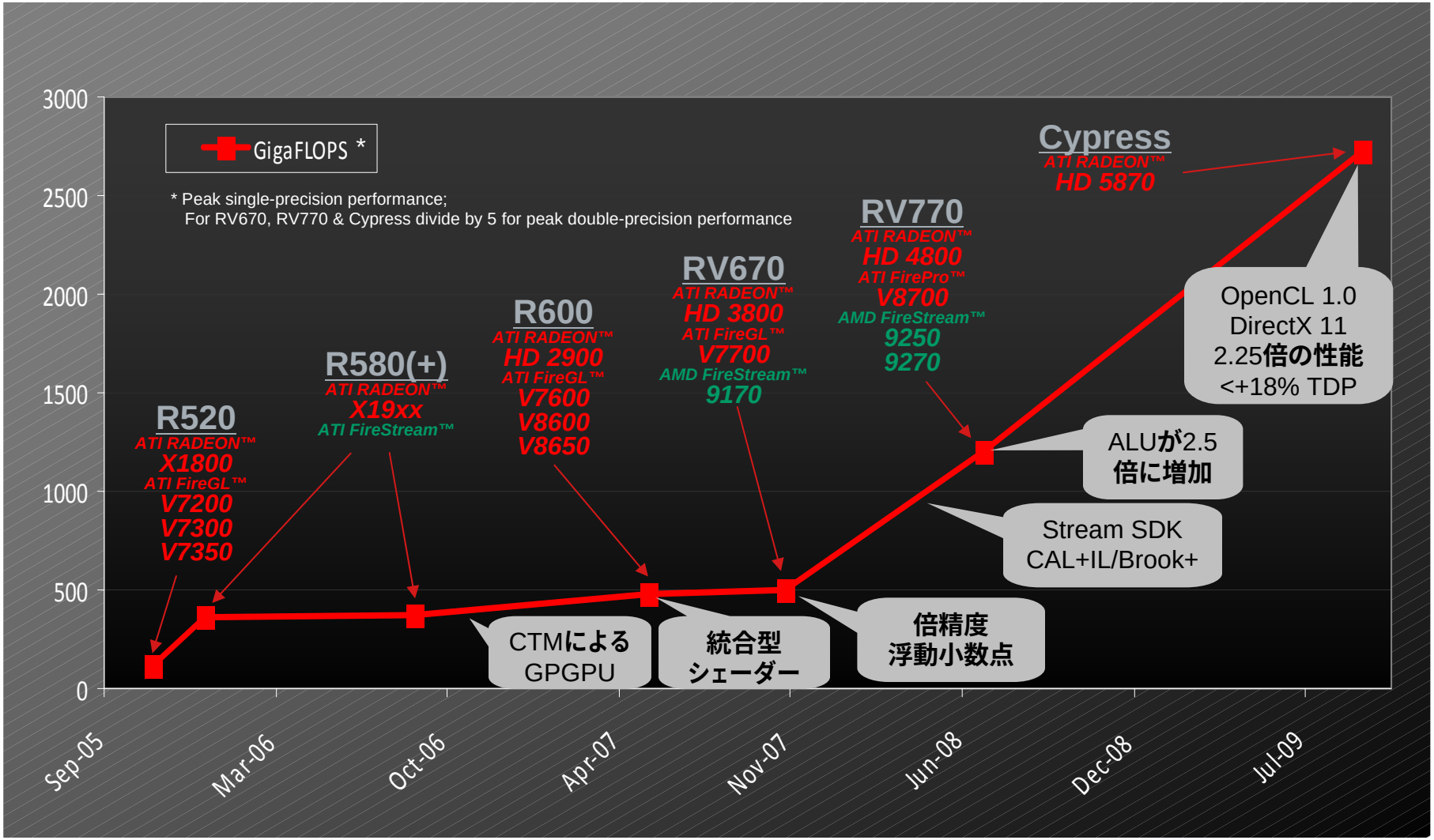


OpenCL™: 融通の利く、拡張可能なオープンスタンダード



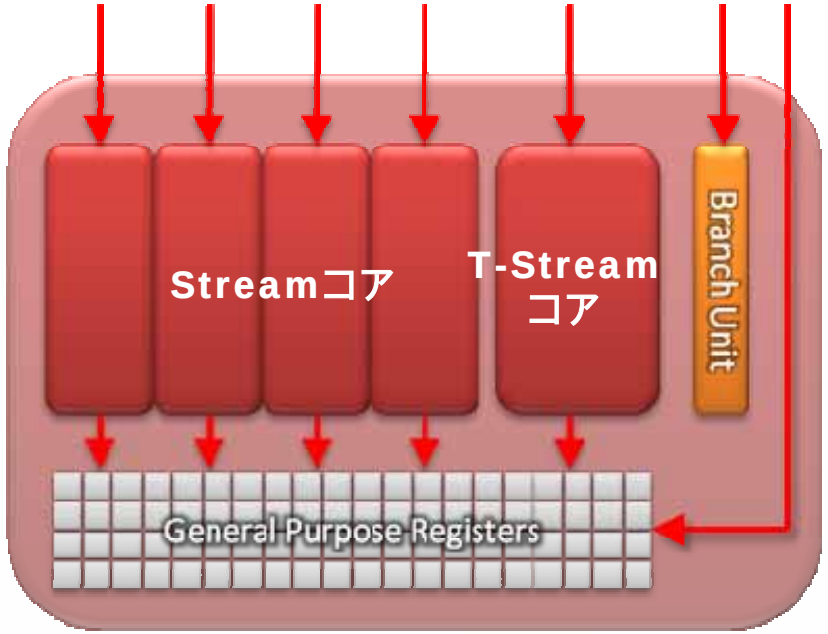
最近のGPUトレンド
- AMD/ATI GPUを例に -

AMD / ATI GPUとStream Computing環境の変遷



AMD / ATI GPUの演算ユニット構成 ー スレッドプロセッサ

- AMD/ATI GPUは多数のスレッドプロセッサで構成されている
  - 5-way VLIWプロセッサ 別名：Shader Processing Unit、パイプライン
  - VLIW命令により、最大5つのスカラ処理を同時に実行
  - 1つ1つのALUをSPU(Stream Processing Unit)もしくはStreamコアという
  - すべてのStreamコアが単精度浮動小数点及び整数命令を実行可能
  - 5番目のStreamコア(T-Streamコア)は超越関数処理もサポート
  - 倍精度演算は4つのStreamコアを組み合わせてサポート(T-Streamコア以外を使用)
  - 分岐ユニットが分岐命令をハンドル

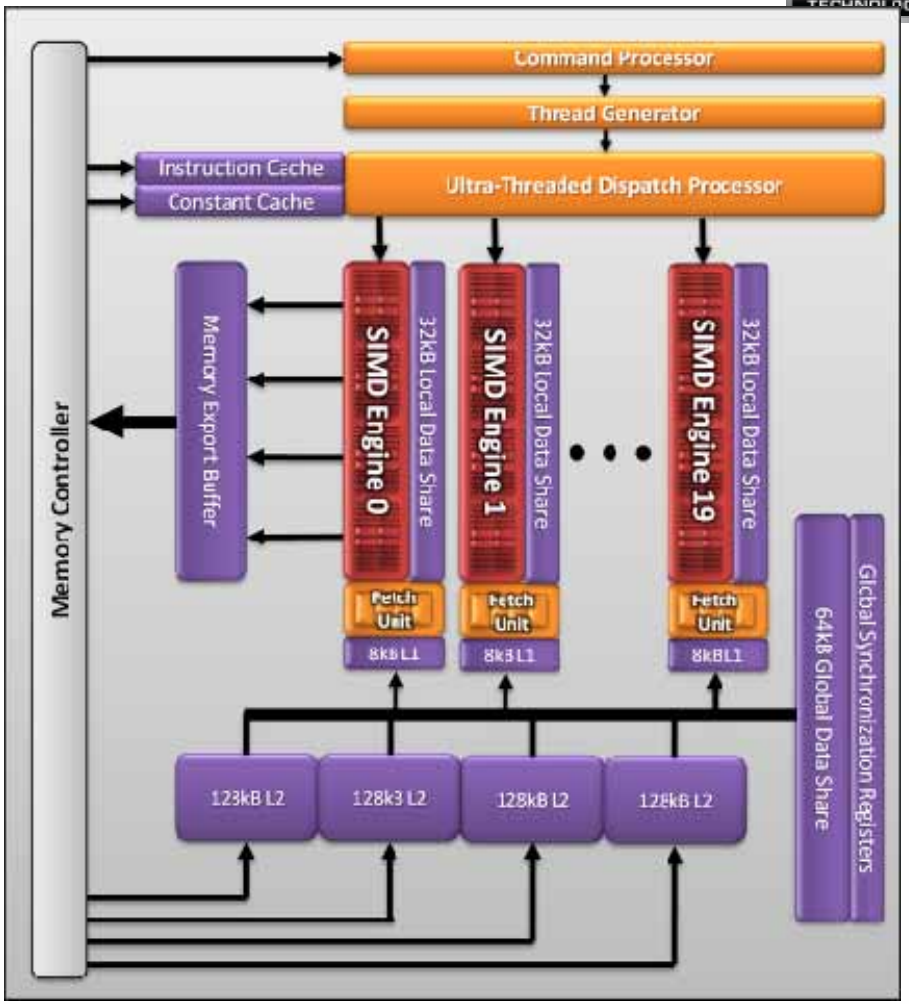


Radeon HD5870のディスアセンブリコード表示例

```
x:  MULADD_e    ____,  R5.y,  R2.x,  T0.x   VEC_120
y:  MULADD_e    R0.y,  R5.y,  R2.w,  T1.w   VEC_120
z:  MULADD_e    T1.z,  R6.y,  R2.x,  T1.z   VEC_210
w:  MULADD_e    ____,  R7.y,  R2.x,  T0.w   VEC_021
t:  MULADD_e    T0.z,  R6.y,  R2.z,  T2.z   VEC_120
```

AMD / ATI GPUのアーキテクチャ － SIMDエンジン

- SIMDエンジン
  - 各SIMDエンジンは複数基のスレッドプロセッサを内蔵（ハイエンド製品では16基）
  - 同じSIMDエンジン内のスレッドプロセッサは同一の処理を行う
    ・・・そのため、SIMDエンジンと呼ぶ
  - デバイス全体では複数基のSIMDエンジン
    RV770は10基、Cypressは20基



Cypressのアーキテクチャ

GPUとOpenCL用語 – 処理単位とワークグループ

GPUチップ	→	OpenCLデバイス
SIMDエンジン	→	コンピュート・ユニット (CU)
Streamコア	→	プロセッシングエレメント (PE)

同一のCU上で実行される関連するワークアイテムの集まりをワークグループという。ワークグループ内のワークアイテムは同じカーネルを実行し、ローカルメモリとワークグループバリアを共有する。

ワークグループにはユニークなワークグループIDが割り当てられ、ワークグループに属するワークアイテムにはユニークなローカルIDが割り当てられる。
例えば、2次元のインデックス空間のグローバルID (gx, gy)、ワークグループID (wx, wy)、ローカルID (sx, sy) の関係は以下の通り:

$$(gx, gy) = (wx * Sx + sx, wy * Sy + sy)$$

ここで、(Sx, Sy) はワークグループのサイズとする。

詳しくは
The OpenCL Specification, *Version: 1.1*に記述があります。

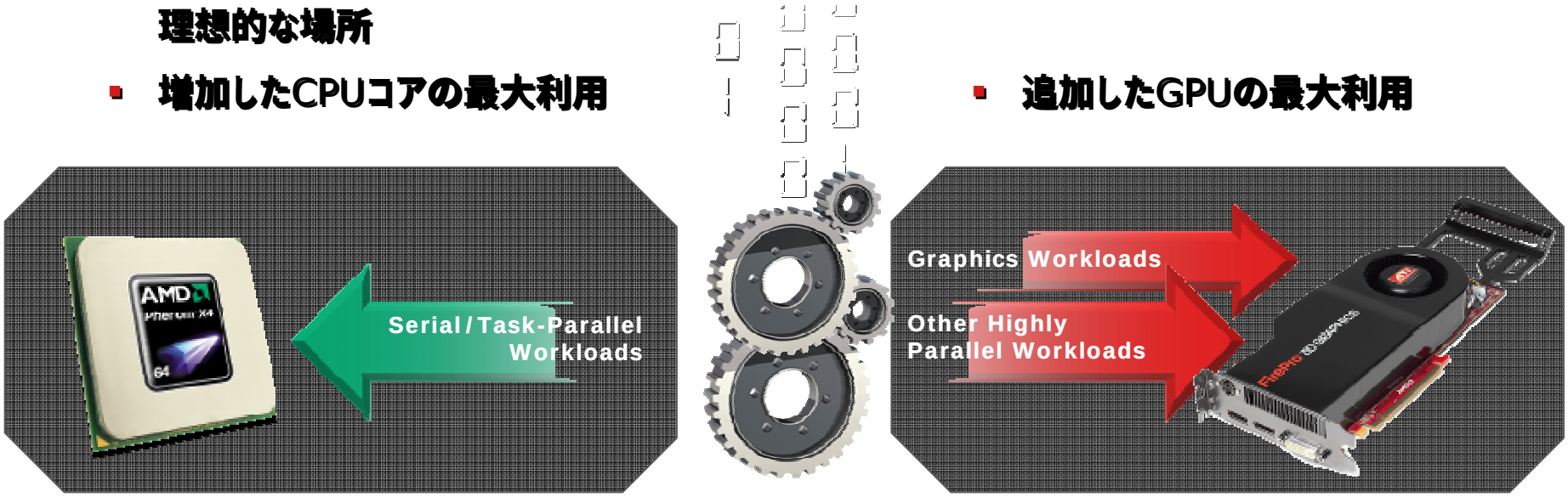
まとめ - バランスの取れたプラットフォームの利点

CPUが得意なアルゴリズムもある
さらに…

- **GPUがフルに演算を行っている場合、CPUはさらに処理を行う理想的な場所**
- **増加したCPUコアの最大利用**

GPUは、画像処理やCAEのような
データ並列化アルゴリズムの処理が得意

- **GPGPU機能の最大利用**
- **追加したGPUの最大利用**



幅広いプラットフォーム構成で最適な性能を確保

Disclaimer & Attribution

DISCLAIMER

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION