PCI expressを進化させた次世代インターコネクト規格CXLの概略紹介

2022/10/19(SNIA), 2022/10/24(NEDIA)

富士通株式会社

Linuxソフトウェア事業部

五島康文

自己紹介

FUJITSU

- 五島康文

  - Linuxの開発部門で、Linux Kernelやその周辺のOpen Source Software(以後OSS)の機能開発を担当
    - ソフト開発が専門（ハード開発部隊は別部門）
    - OSSを弊社向けに独自拡張するのではなく、OSSそのもの(upstream)を標準として作り変えるのがミッション
      - 変更パッチのOSSコミュニティへの投稿、議論、当該パッチの標準へのマージがお仕事
        - ビジネスとしてupstreamへの機能のマージが必須
          - 独自拡張したOSSはその後のメンテナンスコストが高すぎて、結局割に合わなくなる
    - 2003～2008頃、弊社サーバ(PRIMEQUEST)向けにLinux KernelのMemory Hotplug機能を開発
      - ACPIなどの標準化された仕様を使って、弊社独自の機能をLinux Kernel標準のソースで実現
    - 2016/7以降から現在まで、開発チームを率いてLinuxの不揮発メモリDIMM(NVDIMM)対応を担当

はじめに

FUJITSU

- 誤りがないよう努力していますが、CXLについて筆者が誤解している個所や不正確な個所がないとは言い切れません
  - CXLおよびその関連仕様書はかなりのページ数であり、筆者もまだまだごく一部の箇所しか読めていません
    - CXL 3.0仕様は911page (CXL 2.0の時は628pageで+283page)
    - CXLの仕様の前提となるPCIeのGen5 仕様本体で1299page
    - ACPI ver.6.4で1063page
  - 仕様の原文を確認するようにしてください
    - 公式サイトからどなたでもダウンロードできます。(名前とメールアドレスの登録は必要)
    - 誤りがあれば、ご指摘いただけると幸いです
- 仕様が難しいと感じたらslideshare公開済みの資料もご覧ください
  - PCIeやACPIなどの前提知識が無いと、読み進めるのは難しいです
  - 略語が非常に多いので”1.2 Terminology/Acronyms”だけ別途印刷して読むのがおすすめ

今日の内容

- CXLの概略
- CXL 3.0の新仕様
- LinuxのCXL対応状況

FUJITSU

CXLの概略

CXLとは

FUJITSU

- デバイスなどを接続する新たなインターコネクトの規格
  - Compute eXpress Linkの略で、“Open industry standard for high bandwidth, low-latency interconnect” とされている
  - 現在主流のインターコネクトであるPCI express(以後PCIe) の仕様を流用 (PCIe Gen.5以降が条件)
  - 特にGPGPU, Smart NIC, FPGA, メモリ(揮発性、不揮発性), Computational Storage のようなデバイスの接続でメリット
  - 最新仕様はv3.0で、2022/8/1付けで仕様が公開
- 競合していた仕様の中では、生き残りが確実な情勢
  - CXLのBoard of Directorの企業 (2022年8月調べ：赤字は2月以降に参入)には、主要なベンダーがそろう
    - Alibaba, AMD, Arm, CISCO, DELL, Google, HPE, Huawei, IBM, Intel, Meta(facebook), Micron, MicroSoft, nvidia, SAMSUNG
  - CXLと目的が近いライバルの仕様としては、CCIX、OpenCAPIが存在したが、趨勢は決まった感
    - Open CAPIは2022/8/1にCXLに吸収されたことが明らかにされた
    - CCIXのPromoter企業も、現在はそのほとんどがCXLのメンバー企業となっており、CXLに事実上乗り換えたとみられる
      - CCIXのPromoterとして参加していたのはAMD, ARM, HUAWEI, Mellanox(現nvidia), Qualcomm, XILINX(現 AMD子会社)
      - 上記のうち、Qualcommを除く全企業がCXLのBoardとなっており、そのQualcommもCXLのContributorメンバーとして参加
      - CCIXのPress Releaseは2019年を最後に新たな情報なし
  - システム外の接続規格としてGen-Zもあったが、このコンソーシアムも2021年に吸収済み

CXLが生まれた背景

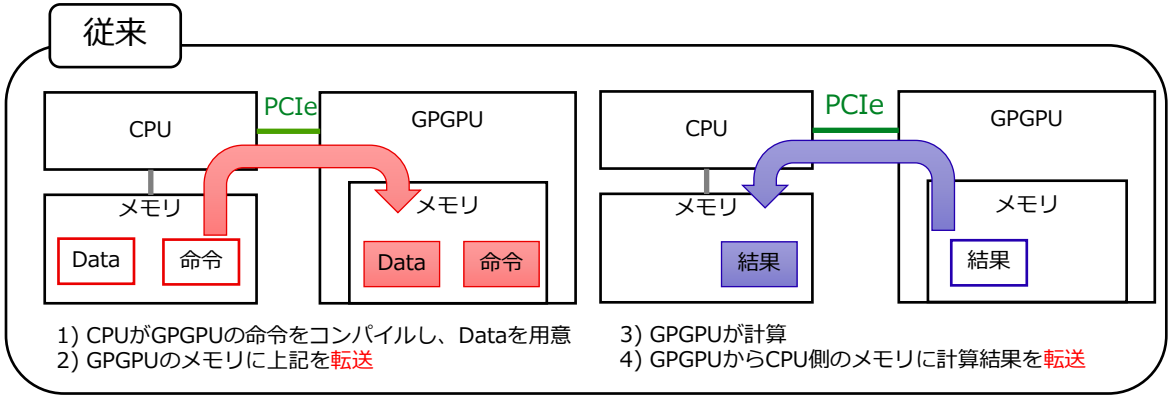
FUJITSU

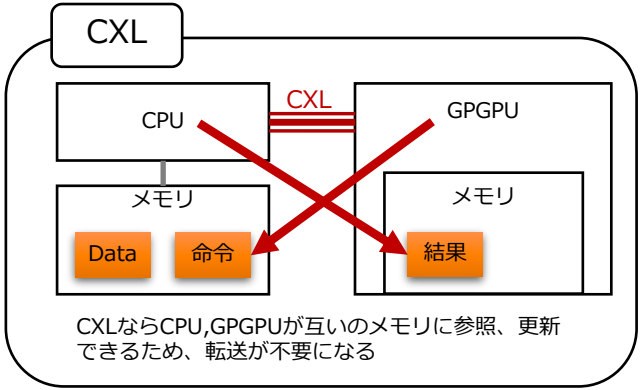
- データの高速な処理の必要性の増大
  - 機械学習などの現在の技術トレンドの影響
- CPUの処理速度の頭打ちによる処理のオフロードの必要性
  - GPGPU、FPGA、Smart NICなどによるデータ処理の肩代わり
- メモリの容量の増大の必要性
  - CPUピン数の制約などによりパラレルインタフェースの DIMM(だけ)では限界

PCI expressに代わる、デバイスやメモリを接続する
新たなインターコネクトが必要

CXLによって何が嬉しいのか？

- 例) GPGPUなどの計算が効率的に行える
  - 現状では、CPUが使うメモリ(DRAM)とGPGPUの”デバイス内の”メモリ間で、互いにデータ転送が必要(*)
    - 計算対象のデータだけでなく、GPGPUに実行させる命令の転送も必要で、面倒なうえ時間がかかる
  - CXLではCPU,GPGPUが、お互いのメモリをinteractiveに直接参照・更新できる
    - 機械学習などの計算が効率よく行えると考えられる
  - FPGAやSmartNICによるデータ処理のオフロードについても同様





- また、DDRxで接続するDRAMだけでは足りなくなってきたメモリ容量の増設なども視野

(*) 参考：CUDA Programming https://www.sintef.no/globalassets/upload/ikt/9011/simoslo/evita/2008/seland.pdf

互いのメモリへのアクセスを効率的に

FUJITSU

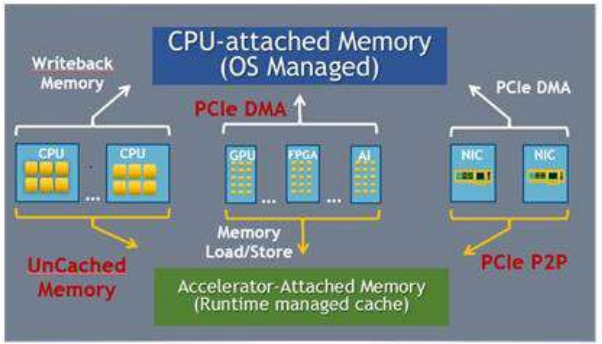
- 互いのメモリをinteractiveに更新するためにはCacheを使ったアクセスが不可欠
  - PCIeではCPUやデバイスのCacheを使ったアクセスはできない
    - デバイスのメモリ空間をホストの空間にマップしてもWrite Through のアクセスしかできず、遅い
    - DMAのような一括したデータ転送を行うことが前提であり、互いにinteractiveなメモリ参照、更新ができない
  - 互いにCacheを使ったinteractiveなアクセスをしたい
    - 必要に応じてWritebackさせたい

CXLは上記を実現する仕様と言える

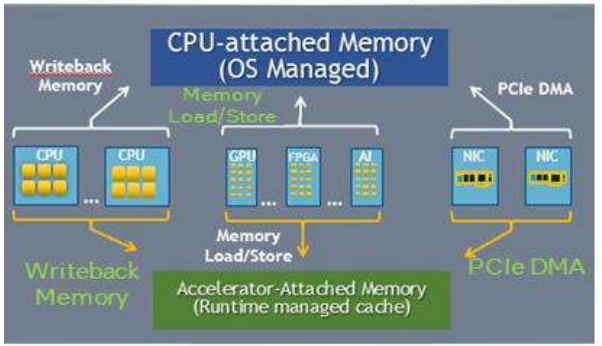
  - 逆に、interactionが少ないデバイスでは、CXLの恩恵は受けにくい
    - データを一括処理してDMA転送し、終わったらCPUに割り込みをかける…とかの程度の機能のデバイスならあまり益はないとされる

右図は以下より抜粋(出典元はIntelのkeynoteの模様)

https://www.servethehome.com/wp-content/uploads/2021/08/Intel-Hot-Interconnects-2021-CXL-1-Open-Interconnect.jpg



With PCIe-only



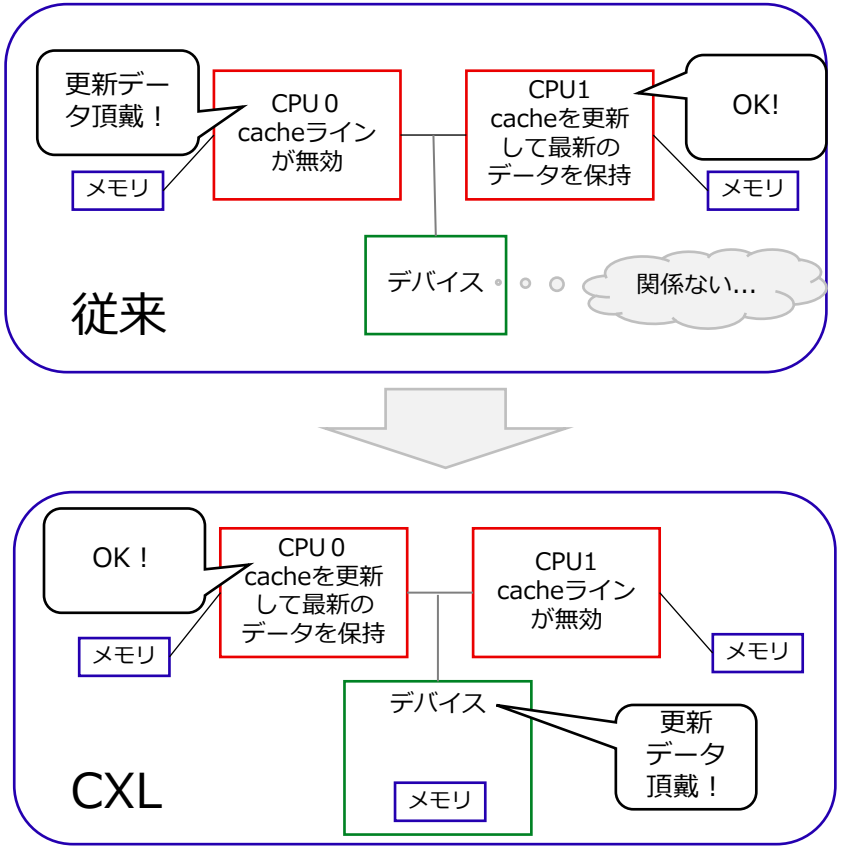
CXL Enabled Environment

Cache-Coherentなinterconnect

FUJITSU

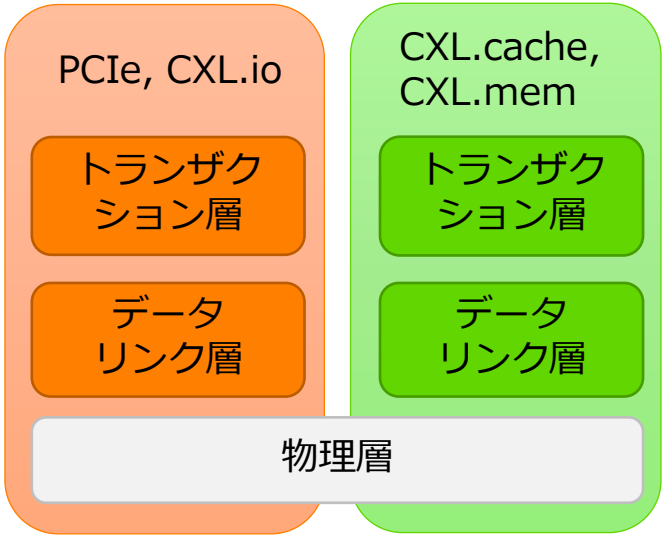
- CXLのWebページ(*)ではCXLを一言で以下のように説明
  - “Compute Express Link™ (CXL™) is an industry-supported Cache-Coherent Interconnect for Processors, Memory Expansion and Accelerators. “
    - ⇒ ここでいう「Cache Coherent とは？」
- これまではCPUの間だけで、メインメモリに対するキャッシュの内容について、互いに調停すればよかった
  - MESI, MOESIなどのcache coherence プロトコルが利用されている
- 前述のCacheでの互いのアクセスを実現するためには、デバイスとCPUの間でも、同様の調停を行う必要
- CXLではMESIプロトコルを採用してこれを実現

状態	意味
Modified	メモリが変更され、当該キャッシュにのみ最新データが存在
Exclusive	当該キャッシュにのみ最新データが存在しているが、メモリとはデータが一致(clean)
Shared	ほかのキャッシュにもデータが存在し、メモリとデータは一致
Invalid	このキャッシュラインは無効



CXLの特徴

- Gen 5.0以降のPCIeを流用（右図）
  - 接続(物理層)はPCIeそのままにしつつ、その上位レイヤをCXL独自のプロトコルを追加することで実現
    - PCIe はGen 5.0以降は、PCIeのバス上で異なるプロトコルの通信を許容
- CXLは以下3種類のプロトコルの組み合わせで実現
  - CXL.io：CXLデバイスの検知・PCIeベースのエラー報告などに使用 (PCIeのプロトコルをそのまま流用)
  - CXL.cache：デバイス（アクセラレータ）とCPU間で主にキャッシュ情報を通信・要求するために利用
  - CXL.mem：CPUとデバイス（アクセラレータ）間でメモリアクセス要求をする際に使用



(*)アクセラレータ:SmartNICやComputational Storageのように、CPUの処理の一部を代わりに処理できるデバイス

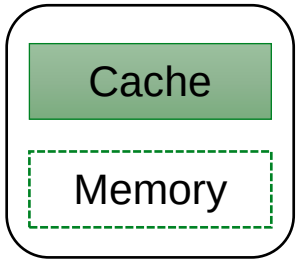
CXLの３種類のデバイス

FUJITSU

- 以下の３種類のデバイスが定義

Type-1

キャッシュを持ちメモリを内蔵しない
or
ホストに内蔵メモリを見せない
デバイス

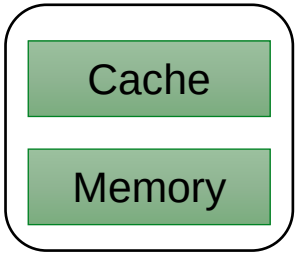


(例：SMART NICやFPGAで該当する構成の場合)

使用するCXLプロトコル
- CXL.io
- CXL.cache

Type-2

デバイスにキャッシュとメモリを内蔵するデバイス

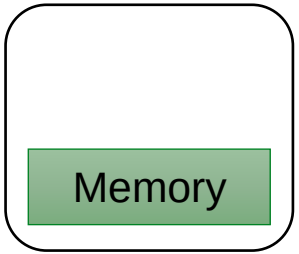


(例：GPGPU、FPGAでメモリを内蔵するケースなど)

使用するCXLプロトコル
- CXL.io
- CXL.cache
- CXL.mem

Type-3

CXLに接続する外付けの拡張メモリ
(揮発性でも不揮発性でもOK)

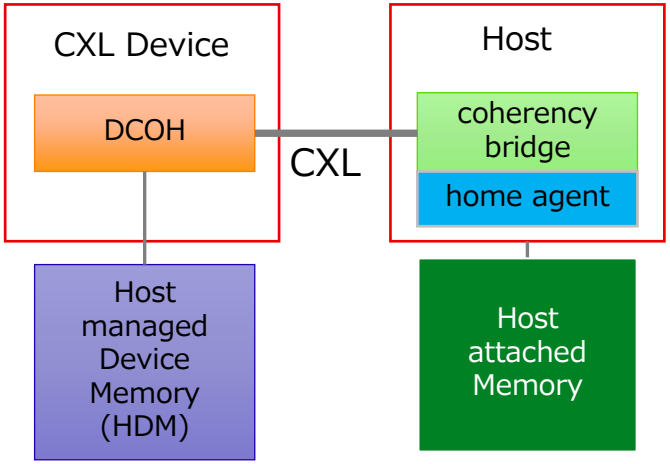


使用するCXLプロトコル
- CXL.io
- CXL.mem

CXL Type2 デバイスのキャッシュとメモリ

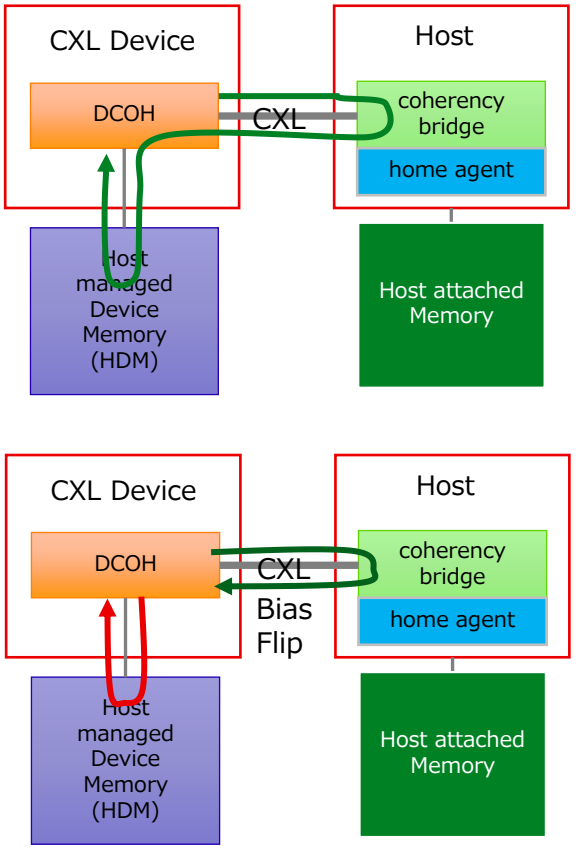
FUJITSU

- Type 2デバイスのDevice側のキャッシュの管理はDCOH(Device's Coherency engine)というコンポーネントが担当
  - Device側のキャッシュの状態管理・更新と適切なメモリアクセス要求を行う必要がある
    - なお、Type3 デバイスの場合はキャッシュの管理はHost側(CPU側)が担当
- デバイスの中のメモリでHost(CPU)に見せるメモリをHost managed Device Memory(HDM)と呼ぶ



Type2 デバイスからのHDMへのアクセス

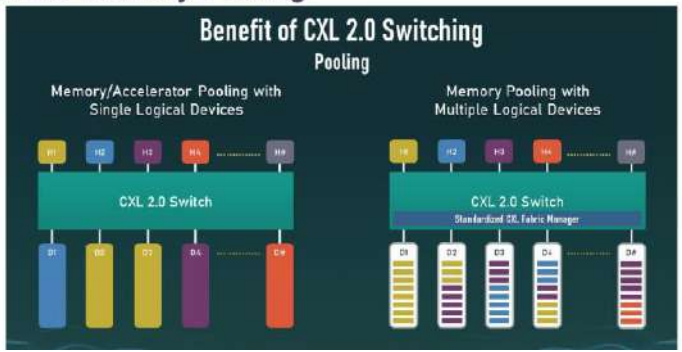
- CXL2.0では、CXL Device側からそのDevice内(のHostに見せる)メモリへのアクセスは、以下の２種類の状態に応じてアクセス方法を使い分ける必要がある
  - Host Bias state（右上）
    - キャッシュのcoherencyを保つためのリクエストをDevice側からCPUに出す必要がある
      - 図の緑の矢印のように、一旦CPU側にリクエストを出してからHDMにアクセスする必要がある
  - Device Bias state（右下）
    - CPU側がキャッシュラインを持っていないことが保証され、Device側は余計なリクエストを送らずにアクセスできる
      - Device Bias modeに切り替える（Bias Flib:緑矢印）と、あとはデバイスが占有してアクセスできる（赤矢印）
- CXL3.0では別の手段が提供(後述)

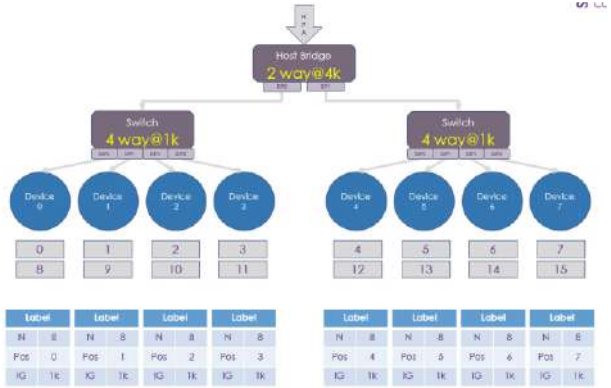


Type 3 メモリデバイスの機能

- 以下のような設定が可能
  - メモリプールとして分割して利用可能（右上）
    - 一つのメモリデバイスをそのまま一つのメモリ領域として提供してもよい
    - 複数のデバイスを束ねて1つのメモリ領域として見せてもよい
    - 一つのデバイスを複数の領域に見せることも可能
  - Interleaveの設定が可能
    - 右下は複数のメモリデバイスをCXL Switchによって4x2=8wayのinterleave設定をした例

CXL 2.0 Memory Pooling

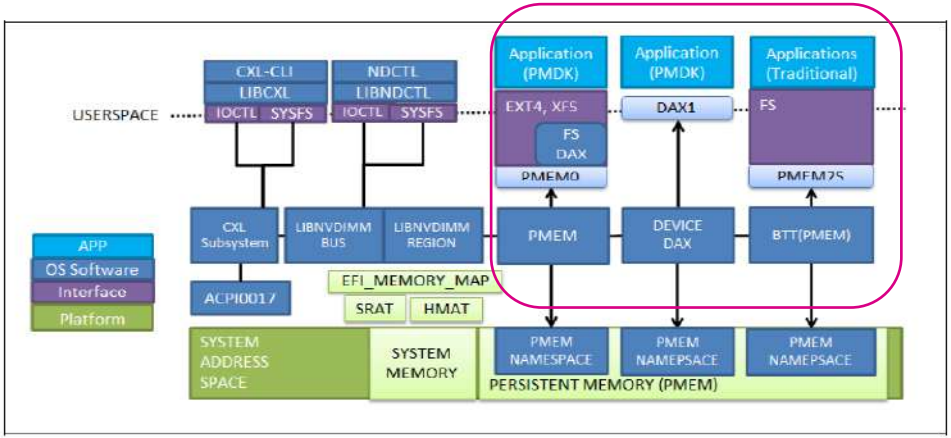




CXL Type 3 メモリのインタフェース

FUJITSU

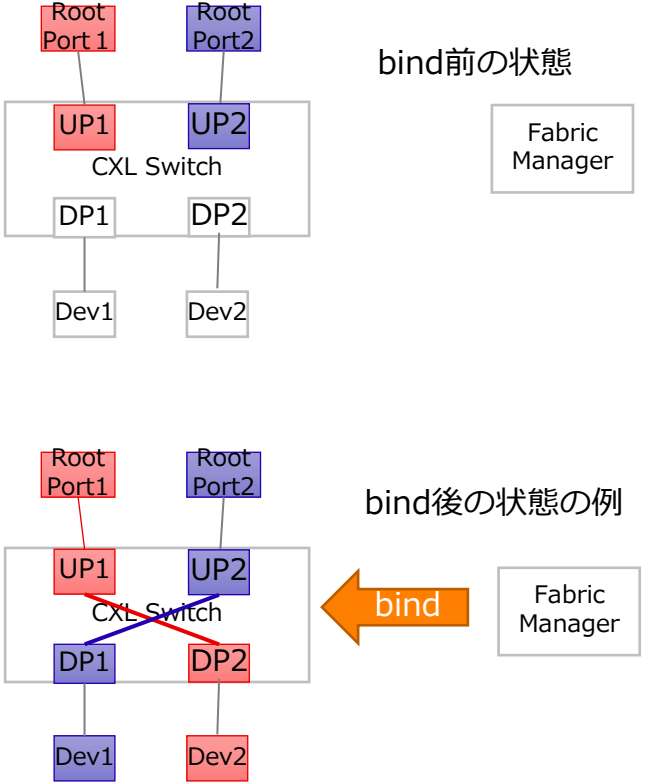
- CXL Type3デバイスは揮発メモリや不揮発メモリを搭載可能
  - CXL2.0 以降では、ドライバをCXL用に新規に作成要
    - ハードの認識・制御方法がDRAMやOptane Pmemのような、DIMMのメモリから変わる
    - CXLのメモリの認識・設定（特にPmem）のドライバは、Linux communityではまだまだ開発中の状態（後述）
- 不揮発メモリのミドルウェア/アプリ向けのIF（右図赤枠内）はOptane Pmemなど、不揮発メモリDIMMのIFが継続
  - Storage, Filesystem DAX, Device DAXといったカーネルが提供する不揮発メモリのIF
  - PMDK (不揮発メモリ向けライブラリ、ツール群)



Optane Pmem向けのソフトがCXL Pmemでもそのまま利用できる見込み

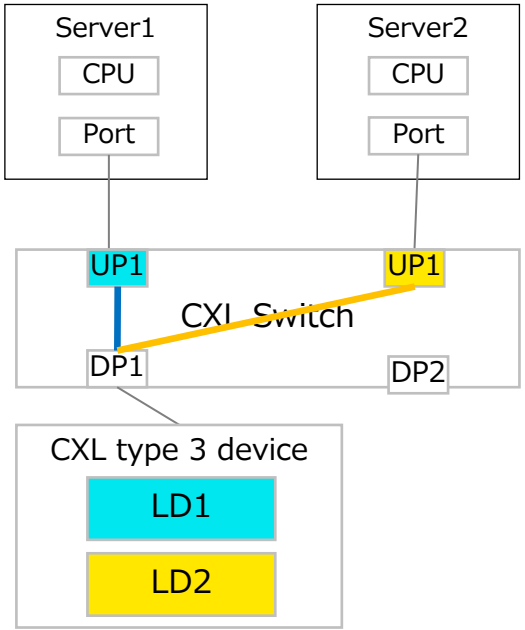
Switchの動的な構成変更

- PCIeではSwitchの上位層(CPU)側のport (upstream port)は必ず１つ
- 一方、CXL Switchは、その中に複数の upstream portを持つことが可能
- 下位(downstream)のportを上位portのどこに bindするかを動的に設定できる
  - ただし、bindの設定するためにはFabric Managerというコンポーネントが存在する必要
    - Fabric managerをどこで実現するかは自由度が高い
      - ホスト上で動作するソフトでもＯＫ
      - BMCで動作する組み込みソフトでもＯＫ
        - サーバー管理ソフトなど
      - CXL Switch内のDevice内組み込みファームウェアでもOK
      - CXL Switch自身による”State Machine”でもOK



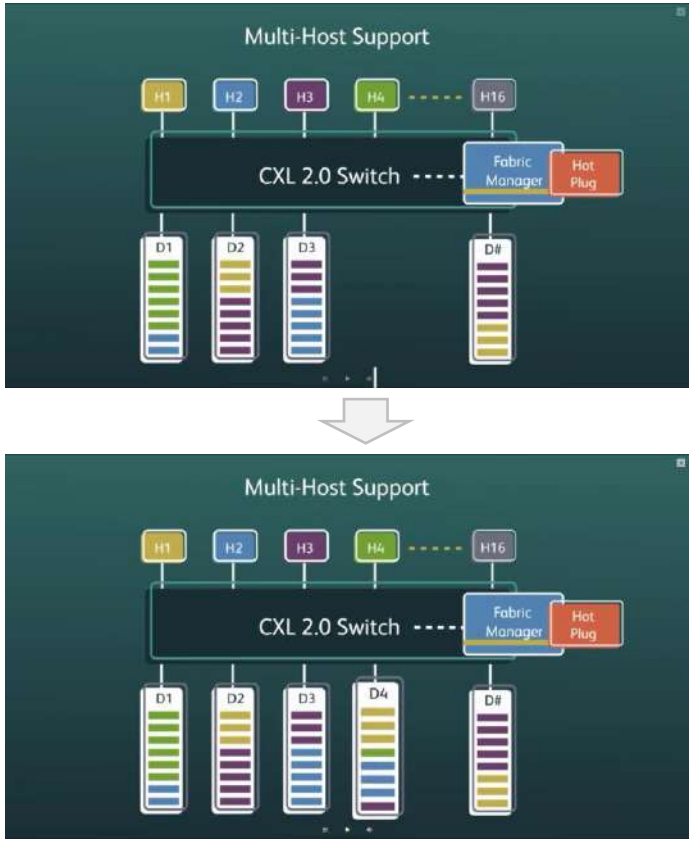
デバイスの論理分割

- Type3 のメモリデバイスはその領域を論理分割し、別々のサーバに分け与えることが出来る
  - 右の図では青の箇所と黄色の箇所を別のUpstream Portに割り当てることが出来る
    - 右図のようにUpstreamの先が別のサーバでもOk
      - 分け与えられたメモリの領域は各サーバが占有
      - 言い換えると、システム間のメモリ共有はCXL 2.0では不可
    - 領域の分割や割り当てを管理についてもFabric Managerの役割



Hotplug対応

- CXL v2.0対応のデバイスは、PCIeのようにHotplug可能なデバイスとなる
  - Type 3のメモリデバイスの場合も同様にHotplugが可能
    - 不揮発メモリだけでなく、揮発メモリについてもハード的にはHotplug可能となる
      - 弊社はPrimeQuestのためにLinuxに実装してきたMemory Hotplugの機能が、CXLにより一般サーバで実現
      - ただし、 Hotplugに関しては、揮発メモリのほうがOSには扱いが難しい（後述)
        - 不揮発メモリの場合、既存のUSBなどのストレージデバイスのつけ外しとほぼ同じ操作
  - 物理的なデバイスの交換のほか、あるサーバーに接続していたメモリを取り外し、別のサーバに取り付けるようなことも可能



CXLを使った製品や予定など

FUJITSU

- CPU側
  - Intelは次期サーバ向けCPU Sapphire RapidsでCXL 1.1(*)をサポート
    (*)不揮発メモリやhotplugなどはまだ未対応の版
  - AMDも次期CPU Genoa/BergamoでCXL 1.1をサポート
  - ArmはNeoverse N2(Arm v9)でCXL2.0をサポート
- デバイス側
  - SamsungはHPC向けにCXLのメモリデバイスを2021/5に発表
    - 合わせて Scalable Memory Development Kit(SMDK)も発表
      - 特定の顧客にのみ公開
    - さらに2022/8のFlush Memory Summitでは、“Memory-semantic SSD“としてCXL接続のSSDを発表
      - 小さなデータの読み書きについて最適化しており、また、ランダムリードでは20倍の性能(latencyとスループット両方?)としている
      - 小さなデータ集合に対する処理が増大するAI / MLのワークロードに理想的とのこと
  - AsteraLabsが CXLメモリコントローラのサンプル出荷を開始
    - DDR5のRDIMMなどを4枚まで挿入可能でPCIe x 16に接続で、最大2TB

CXL 3.0の新仕様

CXL v3.0の新仕様一覧

FUJITSU

- 2022/8/1付けでCXL v3.0の仕様が公開
  - Open CAPI吸収の報と同日
  - 右表はwhite paperより抜粋
  - 個人的な注目点
    - Fabric capabilities
    - Memory Sharing
    - Enhanced coherency
  - それ以外
    - 速度が倍 (PCIe 6.0前提)
    - Switchのカスケード接続 (Switching (Multi-level))
    - デバイス間で直接データ転送 (Direct memory access for peer-to-peer)
    - etc.

Features	CXL 1.0 / 1.1	CXL 2.0	CXL 3.0
Release date	2019	2020	1H 2022
Max link rate	32GTs	32GTs	64GTs
Flit 68 byte (up to 32 GTs)	✓	✓	✓
Flit 256 byte (up to 64 GTs)			✓
Type 1, Type 2 and Type 3 Devices	✓	✓	✓
Memory Pooling w/ MLDs		✓	✓
Global Persistent Flush		✓	✓
CXL IDE		✓	✓
Switching (Single-level)		✓	✓
Switching (Multi-level)			✓
Direct memory access for peer-to-peer			✓
Enhanced coherency (256 byte flit)			✓
Memory sharing (256 byte flit)			✓
Multiple Type 1/Type 2 devices per root port			✓
Fabric capabilities (256 byte flit)			✓

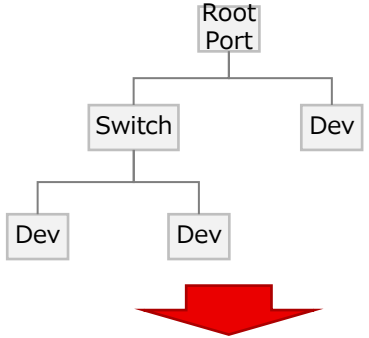
Figure 2: CXL Features over Generations

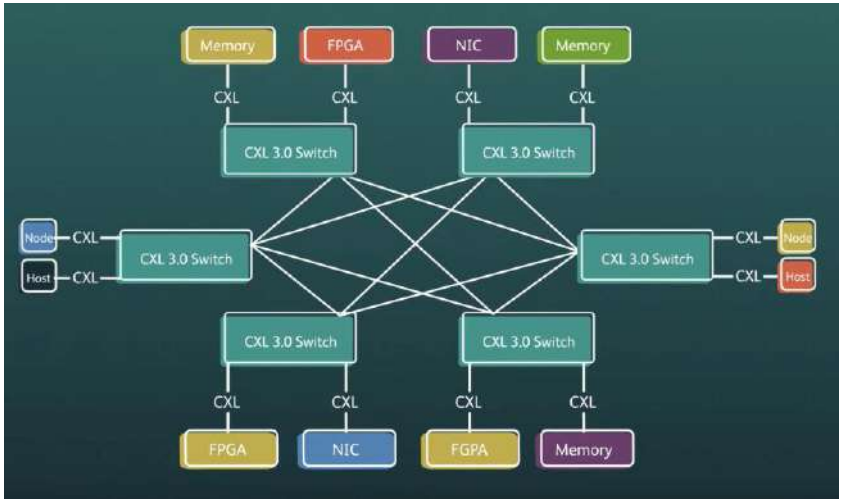
- 今日は個人的に注目している前者３つを説明

Fabric capability (1/3)

- 新たにfabric接続をサポート
  - CXL2.0まではRoot Portを頂点とする木構造
    - CXLでは動的に変更可能とはいえ、木構造は従来のPCIeを踏襲
  - CXL3.0では、右のようにSwitchを介したfabric接続が可能に
  - 最大4096ノードを接続
  - FPGA, SmartNIC, GPGPU、メモリといったCXLデバイスを、サーバー間で自由に最短距離で接続
- 個人的に最も大きな変更点
  - サーバ間接続の在り方が大きく変わる可能性
    - 次世代の分散システムの基盤に
    - かつてのHPEのThe Machine(*)より現実的な印象

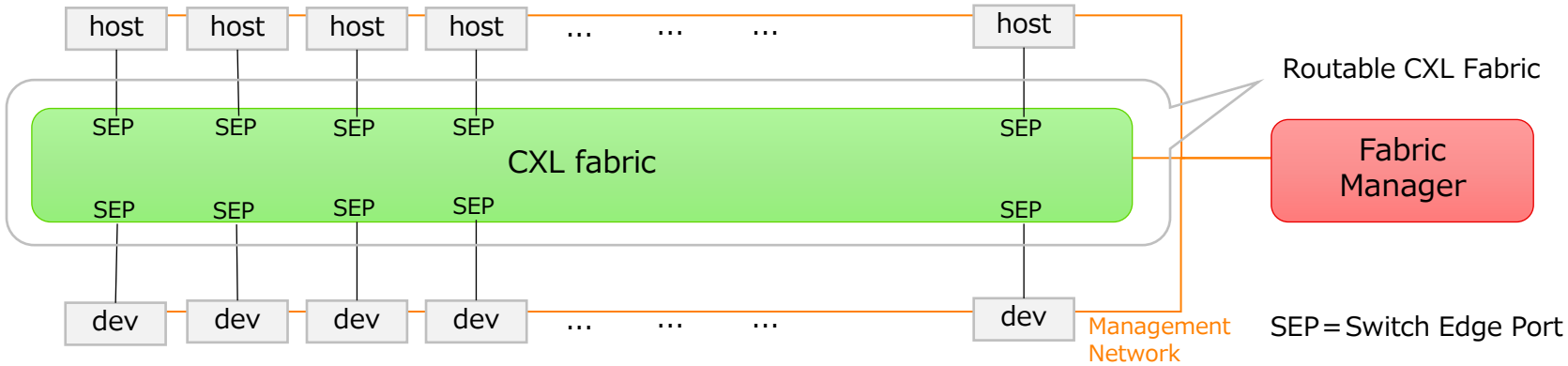
      (*)メモリをシステムの中心に据えるアーキテクチャ





Fabric capability (2/3)

- Port Based Routing(PBR)
  - 送信元や送信先のportのidを指定することで通信
    - idは12bit (4096個分)
    - PBR対応のSwitchはPBR Switchと呼ばれる
    - idの振り分けはFabric manager の役割
      - Management Networkを介して割り当て
        - Management NetworkはSMBus, I2C, I3CでもEthernetでも良いとされている



Fabric capability (3/3)

FUJITSU

- 目についた注意事項
  - 従来のSwitchとの区別
    - 従来のtree構造のみをサポートしたswitchはHierarchy Based Routing(HBR) switchと呼ばれ、PBR switchとは区別される
  - PBRの接続の制約
    - PBRの上位portにはHostのRoot Portのみが接続が許されており、HBRの下位portの接続はできないという制約がある
  - 故障時の回避
    - 故障時の回避ルート選択などの機能は、現段階の仕様では定義されていない
      - できるのはLink Downの検出まで
      - 他にも「これから決める」という記載箇所が散見されるので、Fabric機能の仕様の熟成はこれからかも

Enhanced Coherency(1/2)

FUJITSU

- CXL3.0ではDevice側もCache Coherencyの情報を保持可能
  - CXL2.0ではCPUがこれを主導
    - デバイス側はアクセスする前にCPU側にお伺いが基本

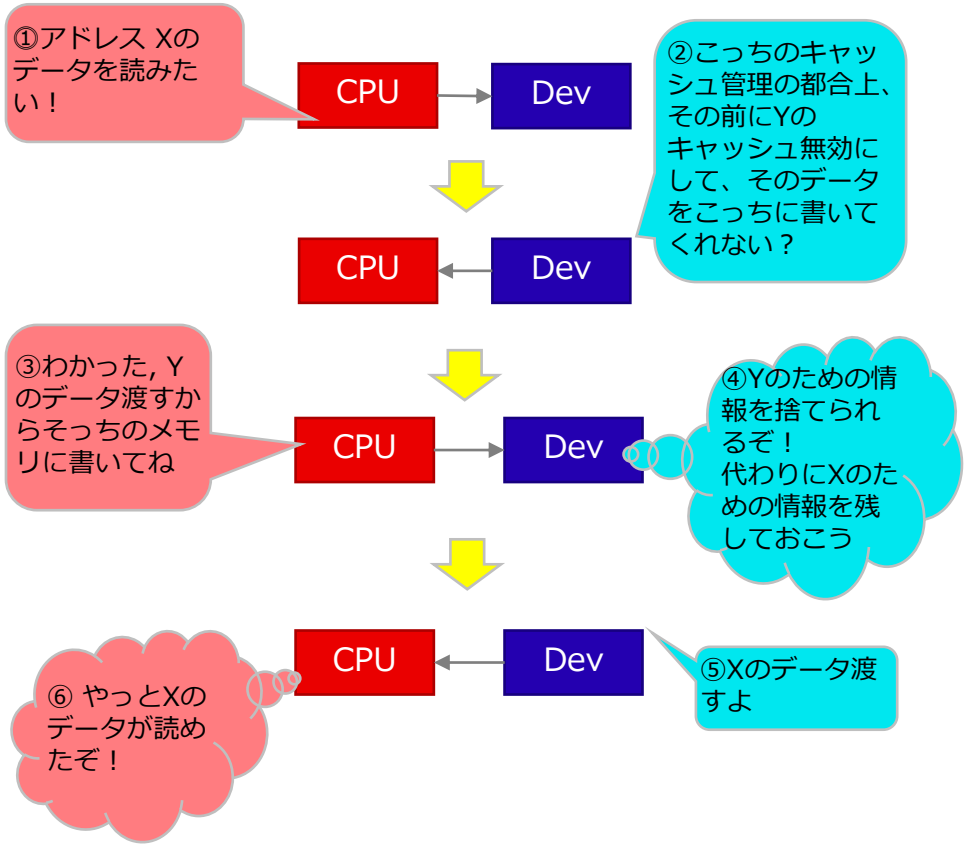
    ⇒ 言い換えると、CPUとCXLデバイスはcache coherenceについて非対称の関係
  - CXL3.0ではデバイス側とCPUの関係が対等に
    - CXL上のキャッシュ更新情報をデバイスのDCOH側でも自ら監視
    - さらに、必要に応じてデバイス側からキャッシュの状態をCPU側に更新するよう要求することが可能に
    - この目的で、CXL.memにBack Invalidation Snoop(BISnp)と呼ばれるチャネルが追加

Enhanced Coherency(2/2)

- BISnpでのやり取りの例
  - 仕様書中には、アクセスパターンやタイミングについて様々なパターンが解説
    - デバイス側の都合でCPU側のキャッシュの状態変更やデータのリクエストを能動的に要求可能(右図)
      - CPU側と連携して互いのキャッシュの状態を管理することがわかる

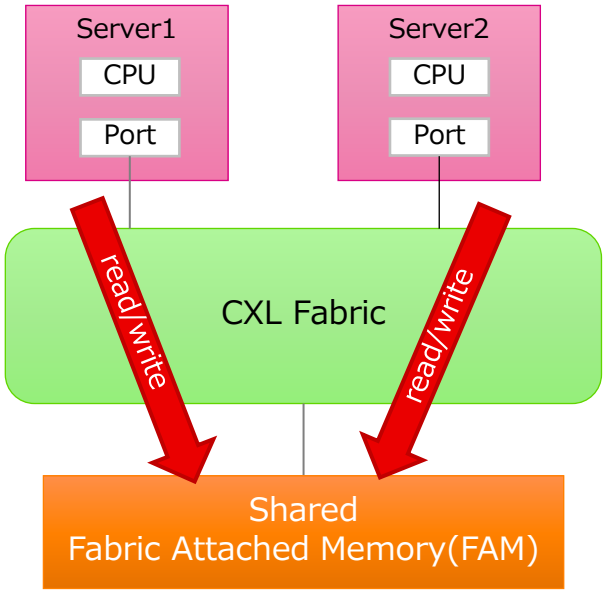
      注）より正確なシーケンスは仕様書の以下を参照
      “3.5.1 Flows for Back-Invalidation Snoops on CXL.mem”
      - 色々なシーケンスが記載
      - 実際のキャッシュの管理がどのように行われるのかが良くわかる

      ⇒ 正直、一読を推奨



Memory Sharing

- サーバー間でのメモリ共有が可能に
  - 互いのサーバが共有したメモリを利用して協調した動作が可能
  - どの領域をどのように共有させるかの設定はFabric Managerの役割
- 共有された領域に対するMemory Coherencyの確保の方法はハードによるものとソフトによるもののいずれか
  - Multi-host hardware coherency
    - 「デバイス側に」Cache Coherencyを追跡できるハード機構を持つ
      - HostのCPU側からのwrite要求に対する他のhostへのキャッシュの調停は、デバイス側が行うことになる模様
  - Software-managed coherency
    - ソフトが自分でCache CoherencyをHost間で調停
      - ハード機構が無い場合でもOK
      - 実際にソフトがどのように調停するかはCXLの仕様範囲外
    - ハード機構が存在する場合も、ソフトがやるかハードに任せるかをソフトが選択可能



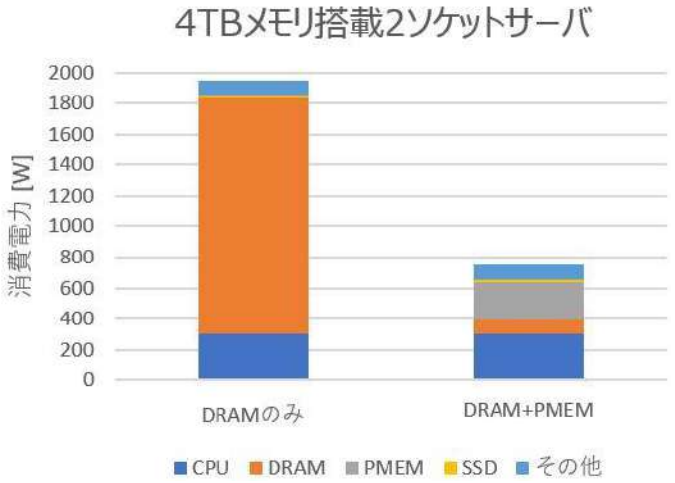
FUJITSU

CXL Type3 メモリの考えられるユースケース

議論の前提としてどのようなことが想定されているか？

大容量メモリ

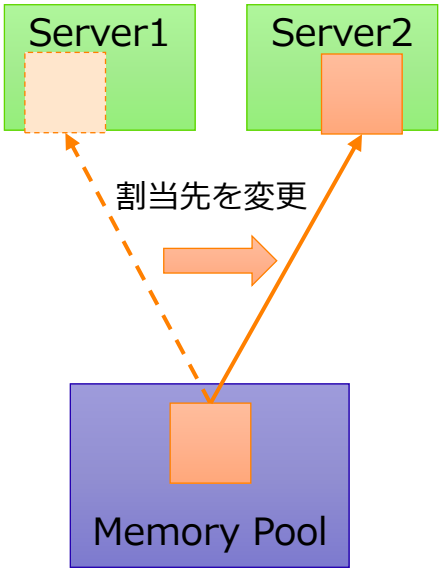
- DRAMは物理的に容量の限界が見えている
  - 物理的な配線の面でも、結線数の多いDDR接続はDIMM数を増やしづらい
    - シリアル接続であるCXLによるメモリ量の増加に期待
  - DRAMのほか、CXL Pmemを大容量メモリとして使う用途は増えると予想
  - MetaはAIのWorkloadが使用するメモリ量の増大に危機感
- PMemの消費電力はDRAMと比べて消費電力が小さい可能性
  - 社内での試算値(右図)
    - IntelのOptane Pmemの公称値(おそらくTDPベース)では1DIMMあたり15W～18W
    - CrutialはDRAMの消費電力を8GBあたり3Wと案内
    - 上記をもとに計算すると、右図のように4TBのような大容量になると消費電力の差がCPUやGPU以上に効いてくる可能性
- NVDIMMと比べるとCXL接続のPMemデバイスの値段は下がると予想
  - PCIeがベースとなっている技術のため、NVMeの技術資産がある程度流用できるかも



(*) カタログ値的な情報を用いた試算値なので、実測値とは異なる可能性があることには注意

Memory Pool

- CXL Type3のメモリを大量に搭載したサーバを用意しておき、必要に応じてメモリが足りない他のサーバ群に分配
  - 銀行でのユースケースの例：
    - 昼：ATMの処理をするサーバにメモリを多く割り当て
    - 夜：給与振り込みなどの夜間バッチ処理をするサーバにメモリを多く割り当て
  - これまでVirtual Machine間ではメモリを融通できたが、実マシンでは不可能だった
  - CXLでは実マシン間での実現が可能
- Memory appliance/Software Defined Memory的な使い方も期待
  - 必要に応じてメモリの足りないサーバにメモリを割り当て
- 先述のようにv3.0の仕様でサーバ間でメモリ共有が可能
  - LFSMMの議論でも、ユースケースとして上がっている
- 共有できないv2.0の範囲でもFailoverとしての使い方も考えられる
  - あるサーバが使っていた領域をfail overでほかのサーバが引き継ぐといった使い方
  - メモリに限らず、将来はGPGPUの処理を途中で引き継ぐといった使い方も考えられる



SLA別の利用

FUJITSU

- 料金の契約別にメモリ割り当てを変更
  - 料金の高い契約をしてる顧客：高速なメモリを割り当て
    - 例えばCPUに一番近い個所のDRAM (*)を割り当て

      (*) 同じNUMA Nodeに属するDRAM
  - 安い契約をしている顧客には、低速なCXLのPMEMや遠い場所にあるCXLのDRAMメモリを割り当て
    - もともとはNVDIMM向けにIntelが作成した機能だったが、CXLでも流用可能

FUJITSU

Linuxの現在の課題やCXL対応状況

Linuxの課題

FUJITSU

- Linuxの対応はv2.0の仕様の範囲でもまだまだ開発中の状態
  - 以下のような課題があり、議論や開発が行われている
    1. CXL v2.0の仕様対応の問題
       - v1.1と比べてハードルが格段に上がった
         - 不揮発メモリの認識設定、CXLスイッチによるinterleaveの設定の認識、hotplugイベントの認識など
    2. Memory Hotplugの問題
       - 揮発メモリのHotplugの扱いは、現在も難しいようでLinuxのイベント(*)でも話題に
         (*) Linux Storage Filesystem| Memory Management Summit
    3. メモリの速度差に対応できていない
       - CXLのメモリではアクセス速度のバリエーションが増える。言い換えるとアクセス速度の階層化される(後述)
       - 現在のLinuxのMemory管理機構が、メモリを階層化を想定していない
         - メモリの配置を最適化できず、性能が悪くなる
    4. etc.
- v3.0の仕様に対するLinuxの対応は、まだまだかなり先になりそう

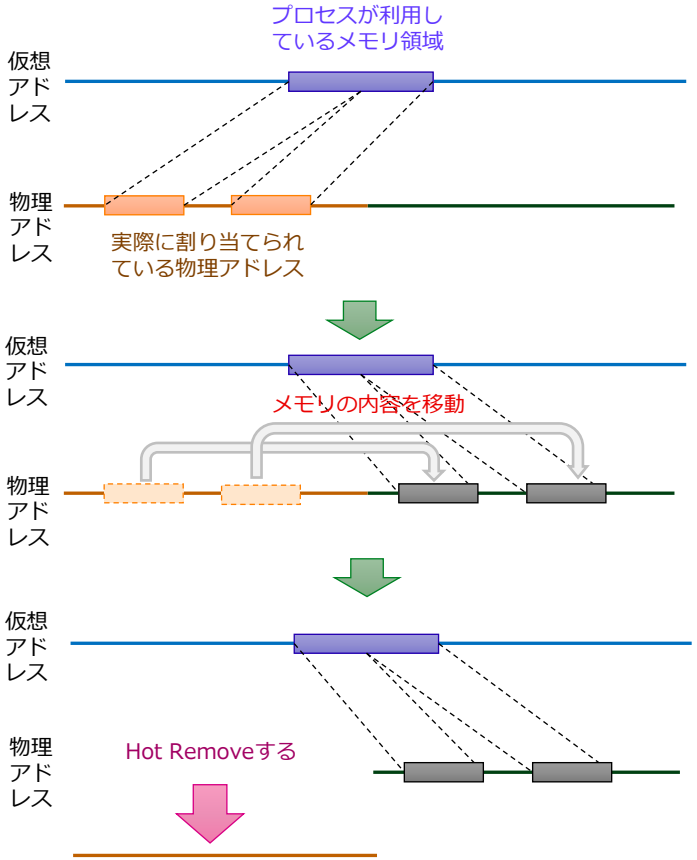
1.CXL v2.0の仕様対応

FUJITSU

- v2.0でOS/driverの役割が大きくなった
  - v1.1では揮発メモリのみしか仕様に定義されてなかった上、hotplugも非対応
    - システム起動時のファームからの情報(EFI Memory MapやSRAT/SLITなど)で読み取るだけで十分だった
    - 言い換えると、従来のDDR DRAMと同じ形でCXLメモリも認識すればよかった
  - v2.0になり、不揮発メモリ対応やhotplugイベントの認識が必要
    - CXL Switchによってinterleaveされたregionの認識、設定
    - NVDIMMと同じくnamespaceの設定が必要
    - CXLメモリのhotplugイベントの検出やOSへの通知
  - このための基本的なドライバや管理コマンドそれ自体がまだまだ開発中
    - Intelが現在ドライバのパッチ投稿中
      - 上記以外にもCXL関係のファームの情報を読むためのドライバのパッチもあり
      - SamsungやHuaweiはすでにこういうパッチのレビューにも参加

2.Memory Hotplugの問題

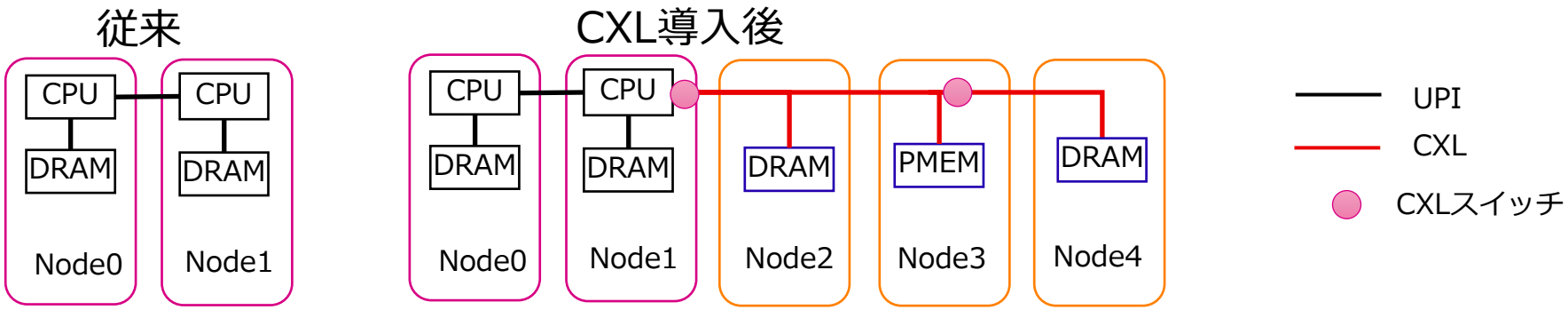
- Memory Hot-removeがうまくいかないケースがどうしても残る…などの問題がある模様
  - MemoryをHot-removeするためには、当該領域のメモリの内容を別のメモリに退避する必要がある
  - ユーザー空間のメモリについては、memory migrationの機能を使って退避し、そのうえでHot-removeする(右図)
    - プロセスの仮想アドレスは変えず、物理アドレスを移動
  - kernel/driverが利用中のメモリ領域は、原則Hot-remove不可
    - Linux Kernel/driverの設計上、移動ができない
    - Hot-removeを想定しているメモリに対しては、kernel/driverが使わないようにあらかじめ設定する必要
  - Linuxのイベントではユーザ空間が使っている領域でも 、結局メモリが抜けないケースがあると話題に
    - 筆者の知る範囲では、Infinibandのようにユーザプロセスのメモリをロックしてしまうような機能もある、少なくともその対処が必要
    - 他は不明だが、Remove時にたまたま何かしらの処理中で移動できないページが残るのかも？



3.メモリ速度の階層化の問題(1/3)

FUJITSU

- CXL接続のメモリは従来とのDRAMとのアクセス速度に差が出る
  - 不揮発メモリも加わり、さらに複数のSwitchをまたがった接続になると、memoryのアクセス速度が階層化(Tiering)していく
    - (CPUと同NodeのDRAM)>(別NodeのDRAM)>(CXLのDRAM)>(CXLのPMEM)>....
- CXL接続のメモリはCPU レスのNUMA Nodeとして扱われる
  - アクセス速度に差がある上、CXLデバイス上でCPUが無いため



3.メモリ速度の階層化の問題(2/3)

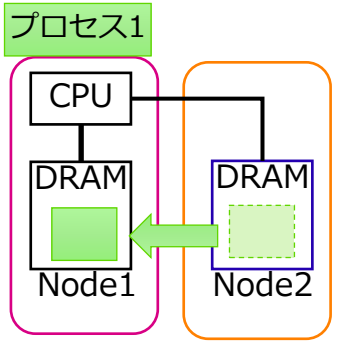
FUJITSU

- 従来のメモリ管理機構がうまく機能しない
  - 階層化、および CPUレスのNUMA Nodeに対する配慮が足りないと問題視
    - Linuxのコミュニティで開発活動が活発
  - Linuxの従来のNUMA balancing機構
    - プロセスが動作するCPUが属するNUMA Nodeのメモリにメモリを移動
      - できるだけCPUと距離が近いMemoryを使う方針の設計
      - 一方CXL メモリはCPU Less NodeになるのでCXLのNodeでプロセスが動作しない

      ⇒ CXLメモリ側にメモリが移動しない

      ⇒ CXLのメモリが活用されにくい
  - コミュニティで活発に議論やパッチが投稿され、この問題について興味の高さがうかがえる
    - IntelとIBMが改善提案のパッチを投稿
    - また、Metaも論文の投稿とそれに対応するパッチを投稿

    ⇒ 次ページではIntelとMetaのパッチの内容を紹介



ただ、全体的にはまだまだ議論が必要な段階で、解決には時間が必要

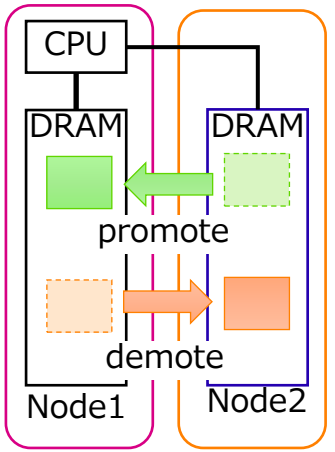
3.メモリ速度の階層化の問題(3/3)

FUJITSU

- Intelのパッチ
  - Swap Out/Inする代わりに、ColdなページをCPU Less NodeにMemoryを移動(demote)させ,Hotになったページを CPUに近いNodeに移動(promote)させる
    - 従来のSwap outの場合、Swap inされるまではプロセスがアクセスできない
    - 一方Demoteの場合、移動後もそのままプロセスが直接アクセスできるのが大きな違い
    - Demoteの判断は既存のページ回収処理をもとに判断
    - Promoteするかどうかの判定は、閾値(default 1秒)以内にアクセスがあったかどうかで判定
    - Demoteは実装が完了したが、Promoteはまだ開発中
- Metaの論文とパッチ
  - 上記だけでは、メモリプレッシャーがある状態（ページ回収処理が動作する状態)でしかメモリの移動が起きない
    - メモリの新規獲得要求が無い状態だと、遠いNodeのメモリを使ったままとなる場合がある

⇒ メモリ回収処理開始の基準となるメモリ使用量の閾値を下げて、より積極的にdemote

  - パッチを投稿した後の動きが見えないのはやや気になる
    - マージされるまでは何度も投稿してレビューを繰り返すプロセスを経る必要があるが、それに気が付いていない？



Linux Plumbers Conference 2022

FUJITSU

- Linux (特にkernel) 開発者向けのカンファレンス
  - referred trackと呼ばれる本セッションのほか、miniconfなどで分野別の話題を取り扱う
  - 主要なLinux kernelの開発者が参加し、発表内容や質疑応答のレベルは高い
    - Linux kernel/driver/周辺libraryの開発者にとって最良のカンファレンス
  - 今年度は、referred trackで2本のCXL関係の発表がなされたほか、CXL専用のminiconf を用意し活発に議論がなされた
    - IntelはTieringの問題を取り上げ、ユーザーインタフェースの提言や、promote/demoteの性能を報告
    - MetaはCXLの全体状況を総括したような、エバンジェリスト的な内容を発表(MetaはBoard Memberでもある)
    - miniconfでは多数の発表があり、盛り上がりを見せていた
      - Armでの開発環境、セキュリティ、CXLSSD、エラーハンドリングなどが話題に
    - 全体的に、CXL対応ハード・ソフトの実現に向けてトライしてみてわかってきた内容となってきた

  ⇒ 昨年と比べると大きく進捗

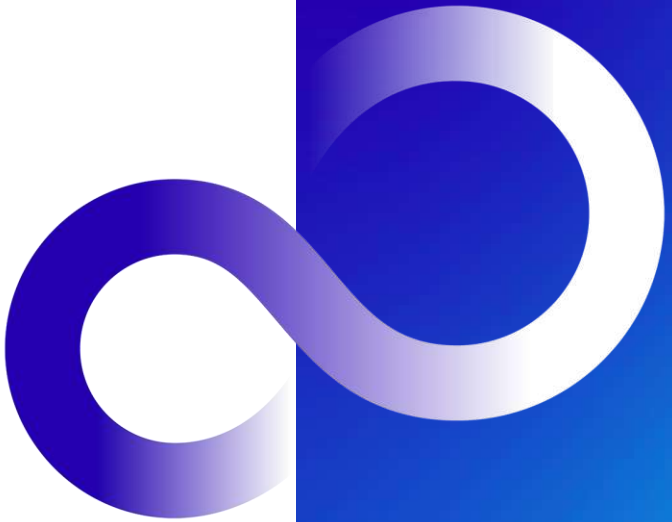
まとめ

- CXLの仕様について、概略を説明した
- CXL3.0の新仕様について説明した
- CXLに対するLinux kernelの開発コミュニティの動向を説明した

日本のベンダーからもCXL対応ハードが多数出て市場が盛り上がることを期待します

「Linuxコミュニティと共に」CXL対応のドライバを作る企業も、増えることを期待します

Thank you



出典資料

- CXL 3.0仕様書
  https://www.computeexpresslink.org/download-the-specification
- CXL 3.0 whitepaper
  https://www.computeexpresslink.org/_files/ugd/0c1418_a8713008916044ae9604405d10a7773b.pdf
- lwn.net
  https://lwn.net/Articles/894598/ および https://lwn.net/Articles/894626/
- CXL* Type 3 Memory Device Software Guide
  https://cdrdv2.intel.com/v1/dl/getContent/643805?wapkw=CXL%20memory%20device%20sw%20guide
- SDC 21 “Compute Express Link™2.0: A High-Performance Interconnect for Memory Pooling”
  https://www.snia.org/sites/default/files/SDC/2021/pdfs/SNIA-SDC21-Rudoff-CXL-Interconnect-Memory-Pooling.pdf