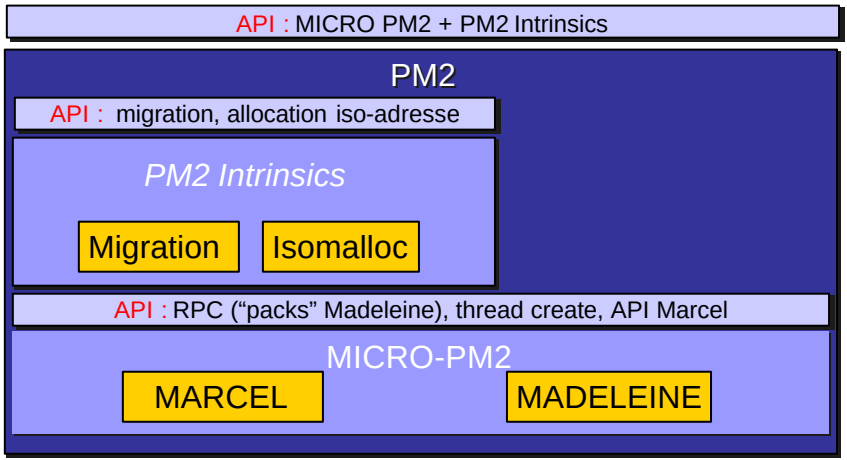
DSM-PM2 : une plate-forme portable pour l'implémentation de protocoles de cohérence MVP multithread

Gabriel Antoniu
LIP, ENS-Lyon

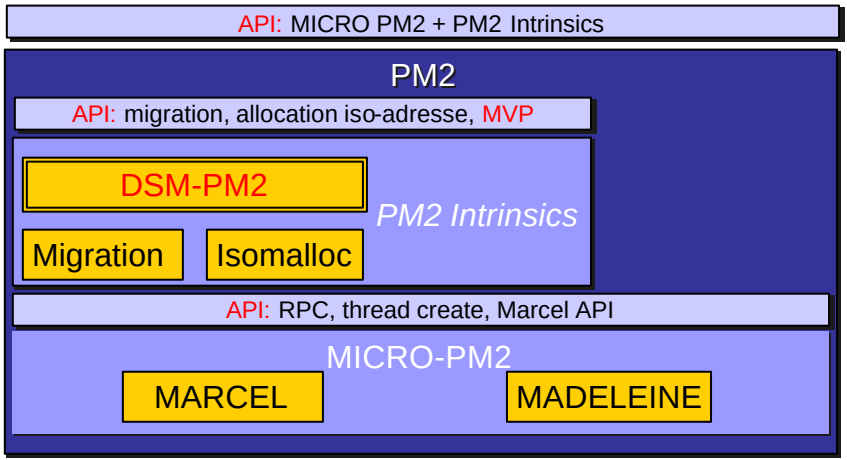
Contexte : PM2

- Environnement de programmation multithread distribué
  - Développé au LIP
- Modèle de programmation : RPC
- Portabilité assurée par deux bibliothèques
  - Marcel threads (en espace utilisateur)
  - Madeleine communications
- Supporte la migration préemptive de threads

Architecture de PM2



Architecture de PM2



DSM-PM2 : point de départ

Quelques questions :

- Modèle de cohérence ?
- Multithreading ?
- Efficacité et portabilité ?
- Migration des threads ?
- Utilisation ?

Quel modèle de cohérence ?

- Le modèle de cohérence dépend de l'application
  - cohérence séquentielle
  - cohérence relâchée
  - cohérence Java

Besoin d'une MVP multi-modèle !

Intégration MVP - threads ?

Threads – entités centrales !

- Environnement multithread
  - Accès concurrents à des données globales
  - Concurrence prise en compte dans les protocoles de cohérence
- Implémentation multithread
  - Gestion multithread de la MVP
- Threads mobiles
  - Utilisation de la migration dans les protocoles

Quelle utilisation pour DSM-PM2 ?

- Objectifs
  - Plate-forme ouverte d'expérimentation
  - Cible de compilateurs (Java, OpenMP, ...)
- Besoins
  - Flexibilité, configurabilité
  - Extensibilité
  - Efficacité et portabilité
  - Outils de traçage et profiling

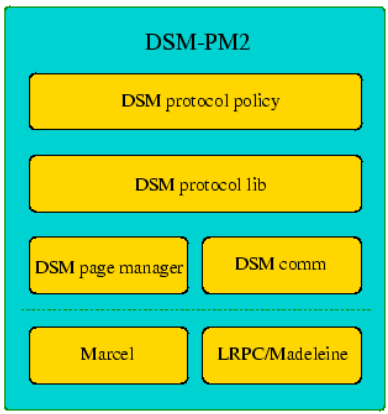
MVP multi-protocoles : conception

- Éléments génériques
  - Communications élémentaires (transfert de pages, invalidations, etc.)
  - Détection d'accès par défauts de pages
  - Table des pages partagées
  - Synchronisation et cohérence
  - Threads !
- Éléments spécifiques aux protocoles
  - Actions pour maintenir la cohérence

Protocole de cohérence

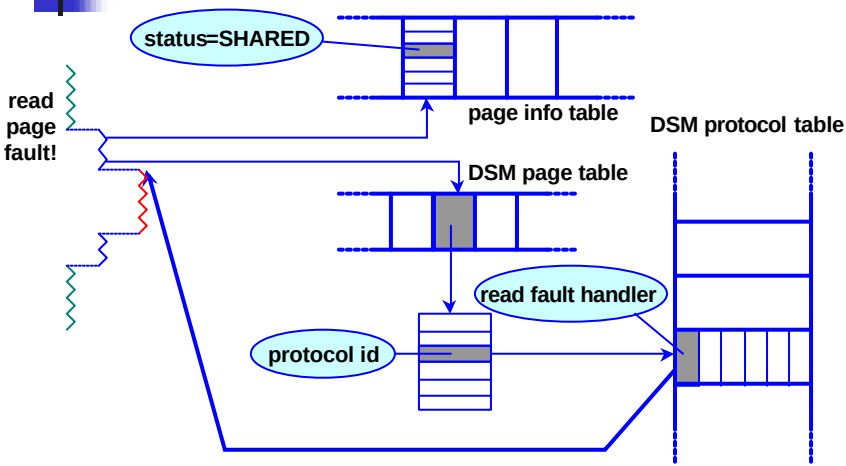
- Protocole = ensemble d'actions
  - Read fault handler
  - Write fault handler
  - Read page server
  - Write page server
  - Invalidate server
  - Receive page server
  - Lock acquire
  - Lock release
- Un protocole implémente un modèle de cohérence

DSM-PM2: Architecture



- DSM comm:
  - ✓ send page request
  - ✓ send page
  - ✓ send invalidate request
  - ✓ ...
- DSM page manager:
  - ✓ set/get page owner
  - ✓ set/get page access
  - ✓ add/remove to/from copyset
  - ✓ ...

Comment ça marche ?

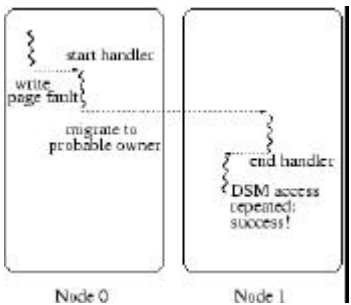


Bibliothèque de protocoles

- Cohérence séquentielle
  - **li_hudak**
  - **migrate_thread**
- Cohérence relâchée
  - **erc_sw** : eager release consistency, single writer
  - **hbrc_mw** : home-based release consistency, multiple writers
- Cohérence Java
  - **java_pf** : Java consistency, page faults
  - **Java_ic** : Java consistency, in-line checks

Cohérence séquentielle basée sur la migration de threads

- Principe: migrer le thread lors d'un défaut de page
  - Possible grâce à l'approche iso-adresse



Utilisation : choix du protocole

- Sélection d'un protocole dans la bibliothèque

```
pm2_set_default_protocol (li_hudak);
```

- Définition d'un nouveau protocole

```
int new_prot;
new prot = dsm_create_protocol ( read_fault_handler, write_fault _handler,
                                 read_server, write_server,
                                 invalidate_server, receive_page_server,
                                 acquire_handler, release_handler);
pm2_set_default_protocol (new_prot);
```

Déclaration des données partagées

- Statique

```
BEGIN_DSM_DATA
int shared_var = 0;
END_DSM_DATA
```

- Allocation dynamique

```
isoaddr_attr_t attr;
int *shared_var_ptr;
isoaddr_set_status(&attr, ISO_SHARED);
isoaddr_set_protocol(&attr, li_hudak);
shared_var_ptr = (int *)pm2_malloc(sizeof(int), &attr);
```

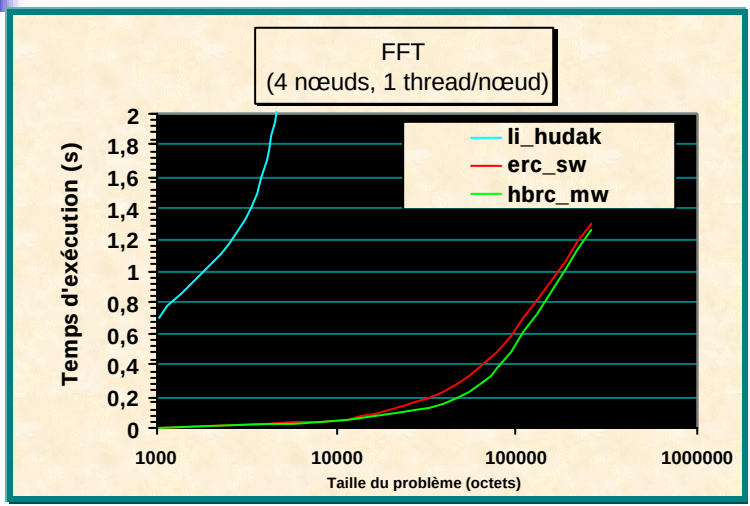
Microbenchmarks (migration des pages)

Opération/protocol	*SISCI/SCI*	*BIP/Myrinet*	*TCP/Myrinet*
Défaut de page	11	11	11
Transmission requête	38	23	220
Transfert de page (4kB)	119	138	343
Surcoût protocole	26	26	26
Coût total accès	**194 µs**	**198 µs**	**600 µs**

Microbenchmarks (migration des threads)

Opération/protocol	*SISCI/SCI*	*BIP/Myrinet*	*TCP/Myrinet*
Défaut de page	11	11	11
Migration thread	62	75	280
Surcoût protocole	1	1	1
Coût total accès	**74 µs**	**87 µs**	**292 µs**

Comparaison des protocoles : SISCI/SCI



Influence du multithreading

- FFT sur TCP/Ethernet (s)

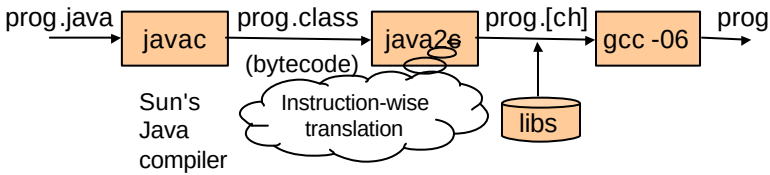
Threads/nœud :	1	2	4	8
li_hudak	74,8	54,8	54,4	57,5
erc_sw	27,2	21,0	19,6	24,1
hbrc_mw	21,4	20,9	20,9	21,1

- FFT sur SISCI/SCI (s)

Threads/nœud :	1	2	4	8
li_hudak	79	57	44	36
erc_sw	1,3	1,3	1,4	1,4
hbrc_mw	1,2	1,2	1,2	1,2

DSM-PM2 : cible d'un compilateur Java pour grappes

- Hyperion (Hatcher, UNH, Durham)



- Exécution parallèle par distribution des threads Java sur les noeuds d'une grappe

Cohérence Java

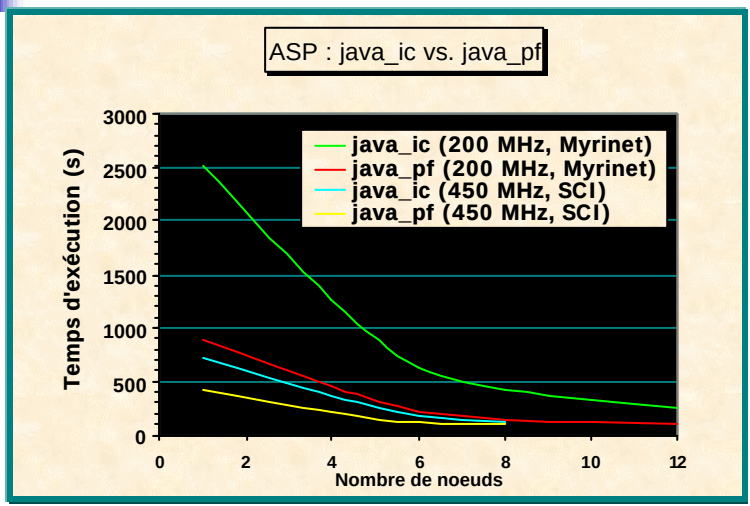
- Variante de la cohérence relâchée
- Les threads peuvent garder des copies des données dans leurs caches
- Règles de cohérence
  - Invalider le cache d'un thread lors de l'entrée en section critique
  - Envoi des modifications locales vers la mémoire principale à la sortie d'une section critique

Cohérence Java dans DSM-PM2

Deux protocoles

- Propriétés communes
  - Home-based
  - MRMW
  - Enregistement des modifications aux objets du cache
  - Envoi des “diffs” au “home-node”
- Propriétés spécifiques
  - Access detection: inline-checks vs. page faults

Java/DSM-PM2 : performances



Autres travaux

- DSM-Threads (Mueller, Berlin)
- Millipede (Schuster, Technion)
- Munin (Carter, Rice)
- TreadMarks (Keleher, Rice)
- CVM (Keleher, Maryland)
- Brazos (Bennet, Rice)

Conclusion

- DSM-PM2 - plate-forme d'expérimentation de protocoles MVP multithread
  - Portable
  - Configurable
  - Co-design application + protocoles
- Cible de systèmes de compilation pour des architectures distribuées
  - Java

Perspectives

- Évaluation plus fine des protocoles (benchmarks SPLASH-2)
- OpenMP sur DSM-PM2
- Protocoles adaptatifs
  - migration des pages vs. migration des threads
- Spécification et de validation formelle de protocoles
- Exploitation de Marcel SMP
- Prise en compte des modèles de mémoire hiérarchiques (grappes de grappes)