DLM (Distributed Large Memory) : Remote Memory Paging for Efficient Use of Memory Resource on Clusters

Hiroko Midorikawa, Seikei University, midori@st.seikei.ac.jp, http://www.ci.seikei.ac.jp/midori/paper/paper.html

User-Level Remote Paging for Multithreaded Applications

DLM offers a virtual large memory using distributed node memories in a cluster for multithreaded applications (OpenMP and pthread programs).

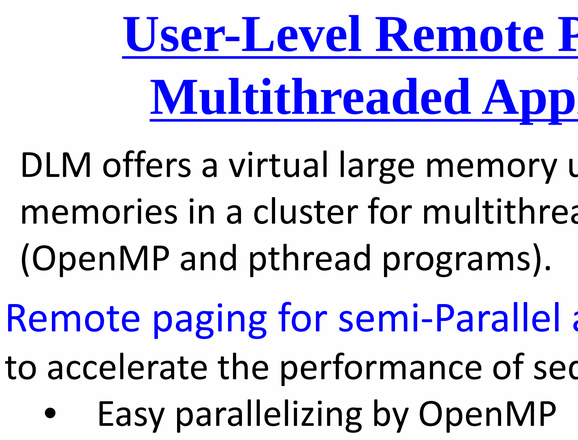
Remote paging for semi-Parallel and Pseudo seq. codes to accelerate the performance of seq. codes

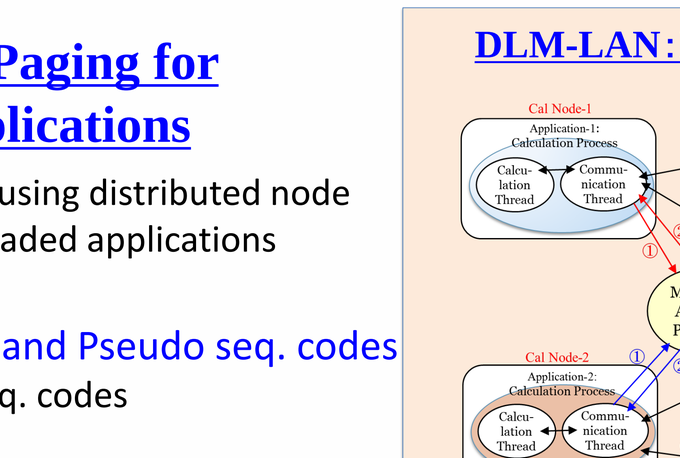
- Easy parallelizing by OpenMP
- Usage of implicitly multithreaded library functions

User threads are dynamically created and destroyed.

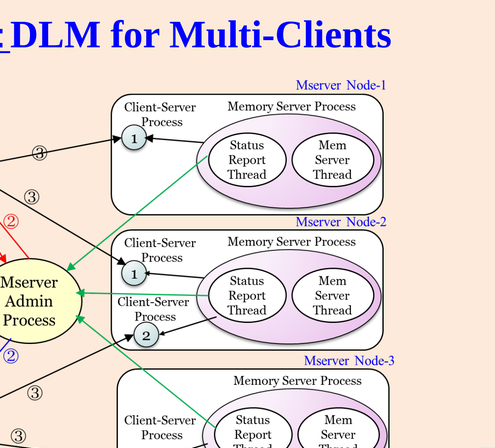
Simple Threads Suspension : user-level implementation

(1)The DLM hooks a pthread_creat() call and replaces it with the tailored pthread_creat() call which registers all user thread IDs.
(2)When a user thread accesses the remote data, the com-thread requests and receives a remote page to/from a memory server in background.
(3)Suspends all user thread with pthread_kill() and copy the page in a receive-buffer to user-space.
(4)Restart all user threads using pthread_kill().





Inconsistent Page problems in user-level remote memory paging

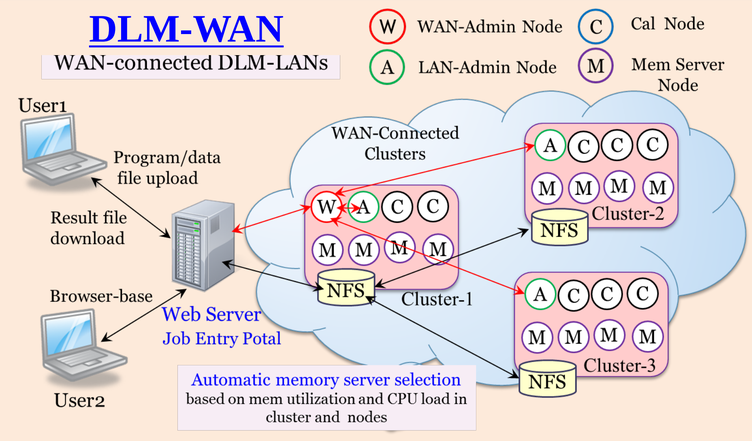


Matrix multiplication(4095x4096)

DLM-LAN : DLM for Multi-Clients

DLM-WAN

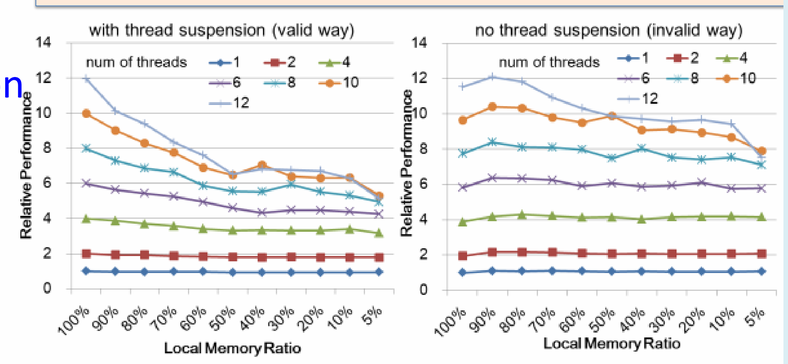
WAN-connected DLM-LANs



DLM Benefits:

- Available to solve bigger size problems using existing sequential codes without local memory size limitation
- Release users from the substantial amount of work
  - *Redesign of the sequential algorithm*
  - *Converting the existing codes to parallel MPI codes*
  - *Parallel code debugging*
- Easy to use remote memory with No knowledge of MPI
- Highly portable to various clusters, user-level software.

Designed for the users who prefer and accept extra execution time caused by using remote memory partially.



Thread suspension overhead in FDA cluster stencil program (8192x8192, 15x15 mask)

Easy to use: only add dlm to existing programs

Specify DLM data that exist in local or remote memory

```
#include <stdio.h>
#define N 16384  // total memory 2^31B + 2^15 B, 2GiB

dlm double a[N][N], x[N], y[N];  // DLM data

int main(int argc, char
{                    :
```

DLM Compiler
dlmcc prog.c -o prog

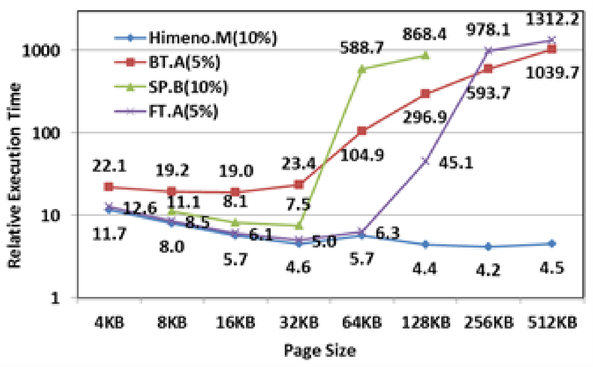
Or use dlm library functions for dynamic allocation

```
#include <dlm.h>
#include<omp.h>
double (*a)[N];  //  a pointer for 2-D array      a[N][N]
double  *x, *y;  //  pointers for 1-D arrays   y[N],x[N]

main(int argc, char *argv[])
{       int i,j;
        double temp;
        dlm_startup(&argc, &argv);

        a = (double (*)[N]) dlm_alloc(  N * N * sizeof(double) );
        x = (double *) dlm_alloc( N * sizeof(double) );
        y = (double *) dlm_alloc( N * sizeof(double) );

#pragma omp parallel for
        for(i = 0; i < N; i++)       //  Initialize array a
          for(j = 0; j <N; j++)    a[i][j] = i;
#pragma omp parallel for
        for(i = 0; i < N; i++)    x[i] = i;     // Initialize array x
#pragma omp parallel for
        for(i = 0; i < N; i++){          // a[N][N]*x[N]=y[N]
            temp = 0;
            for(j = 0; j <N; j++)  temp += a[i][j]*x[j];
            y[i] = temp;
          }
        dlm_shutdown( );
        return 0;
}
```

An example of Compile Command
mpicc prog.c -o prog -ldlmmpi

AAPC: Automatic Adaptive Page-Size Control for Remote Memory Paging

Fixed-size page transmission sometimes introduces drastic performance degradation.

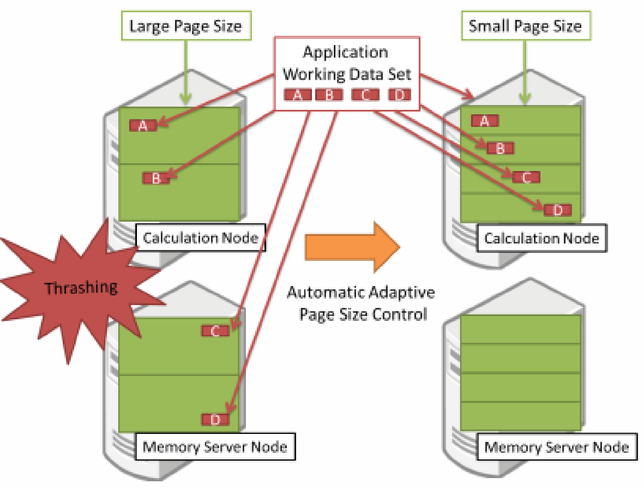


At low ***Local Memory Ratio*** (5%-10%),

- The worst example : NPB BT, SP, FT
  Big page introduces drastic degradation
- Typical example: Himeno Benchmark
  Big page introduces better performance

Local Memory Ratio
= Local memory size / Total program data

It is difficult to determine the optimal page size before program running.



Basic Idea of AAPC

Choose an appropriate page size to depress page thrashing between calculation node and memory server nodes. The AAPC repeatedly estimates a working data set size and changes page size dynamically and adaptively to each processing part of an iterative application during it is running.

Estimation of Working Data Set Size in Applications

2 functions for estimating working data set size for each processing part
- void swapin_countstart (int id);
- void swapin_refresh (int id);

swapin_countstart (id)
- Changes current page size to the target page size of id, which was registered previous iteration.
- Starts new recoding of swapin page

WS estimation is influenced by current page resolution. It is repeatedly estimated with different page size.

swapin_refresh(id)
- Estimates WS based on swapin page records
- Calculates target page size using WS page count & registers the target page size of id
- Resets swapin page records.

Example codes

```
for(ite=0; ite<N; ite++){
  :
  swapin_countstart(0);
    Processing part 0
  swapin_refresh(0);
    :
  swapin_countstart(1);
    Processing part 1
  swapin_refresh(1);
  :
}
```

Typical Execution Command

```
mpirun -np 4 prog args -- –f memfile
```

memfile

Calculation node	24000	24GB for rank0
Memory Server1	20000	20GB for rank1
Memory Server2	10000	10GB for rank2
Memory Server3	16000	16GB for rank3

Memory size/ Node Specification

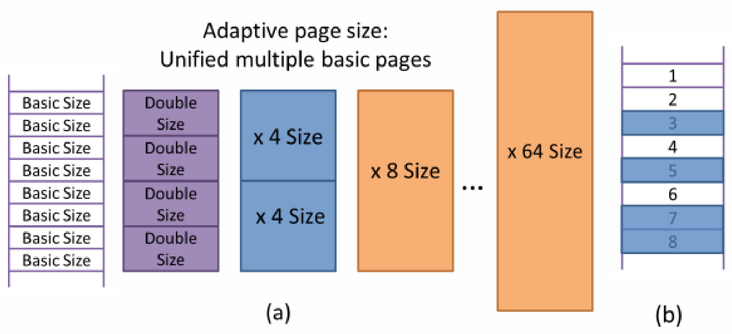
Criterion for Page Size Control

(1) If ***WS Page count*** > ***Local Page count***, a thrashing occurs.
Change page size small as closely as the following ***Target Size***.
Target Size = Local Memory Size / ***WS Page count***

(2) If ***WS Page count*** <= ***Local Page count***, there is no thrashing
Change page size little larger to expect more efficient communication, e.g. double the current page size.

Adaptive Page Size Changing

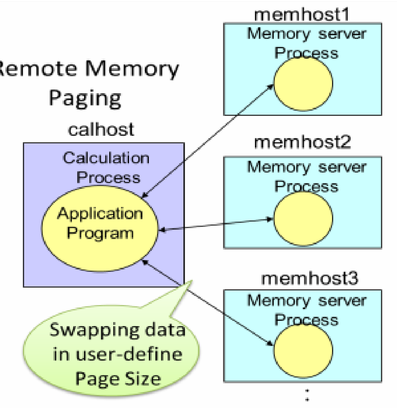
(a)Unified multiple basic pages are used in page size changing
(b)AAPC supports a transmission of a transitional fragmented page generated when page size is changed from small to large
(c)AAPC allows coexistence of different page size transmission.

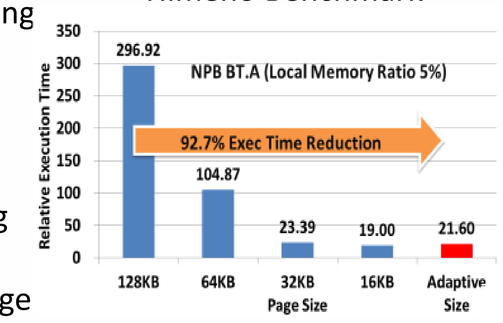


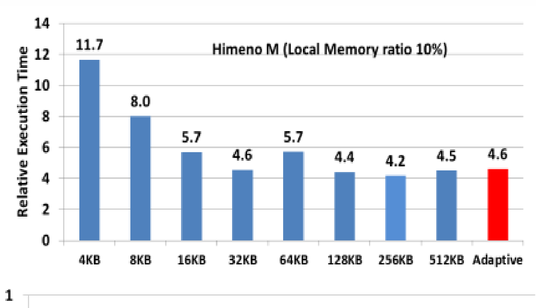
Experimental Results of AAPC

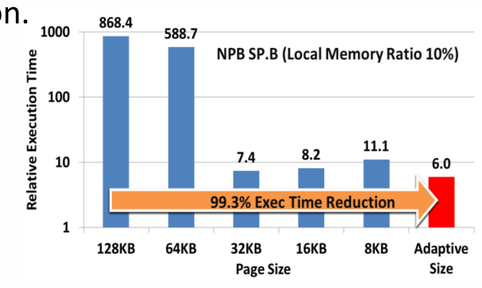
Experiment parameter:

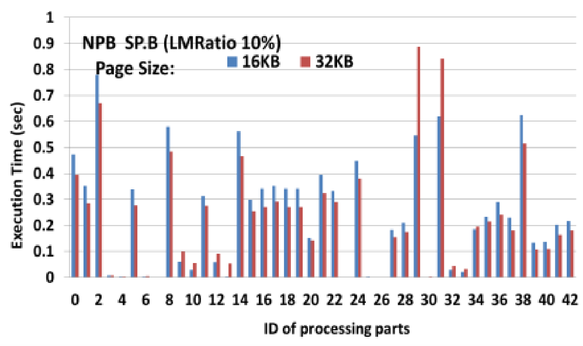
- Start Page size: 128KB
- Page size range: 16KB – 1MB
- NPB: BT, SP, iteration 10
- NPB: FT
- Himeno Benchmark











[1] H. Midorikaw, et.al., "Automatic Adaptive Page-size Control for Remote Memory Paging", IEEE/ACM CCGrid2012, Best Poster Award CCGrid2012