Funzioni PVM	Funzioni WAMM	Descrizione delle routine instrumentate
pvm_exit	wamm_exit	Determina l'uscita di un task da PVM, quando il tracing è attivo provvede alla chiusura ed al strasferimento del file di traccia locale.
pvm_mcast	wamm_mcast	Effettua l'invio in multicast di un messaggio PVM e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.
pvm_mytid	wamm_mytid	Individua il tipo di monitoraggio da effettuare ed inizializza le strutture dati necessarie per mantenere le informazioni sul monitoraggio. Restituisce il TID assegnato al task PVM.
pvm_nrecv	wamm_nrecv	Effettua una receive non bloccante, fornisce il supporto allo *steering* e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.
pvm_parent	wamm_parent	Restituisce il TID del task che l'ha generato, nel caso in cui esso sia WAMM, restituisce il valore PvmNoParent.
pvm_precv	wamm_precv	Effettua una receive ad una fase, il messaggio ricevuto viene copiato nello spazio utente, fornisce il supporto allo *steering* e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.
pvm_psend	wamm_psend	Effettua nello stesso tempo l'impacchettamento e la spedizione di un messaggio e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.
pvm_recv	wamm_recv	Effettua una receive bloccante a due fasi, fornisce il supporto allo *steering* e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.
pvm_send	wamm_send	Effettua una send e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.
pvm_trecv	wamm_trecv	Effettua una receive con timeout, fornisce il supporto allo *steering* e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.

Tabella 1: Elenco delle funzioni PVM che sono state instrumentate.

- Prima di iniziare la fase di monitoraggio, ogni task applicazione deve invocare la funzione `pvm_mytid()`, ciò è necessario perchè la routine che la rimpiazza (`wamm_mytid()`) contiene il codice che inizializza le strutture dati per effettuare il monitoraggio stesso;

- Ogni task per concludere la sua esecuzione deve invocare la `pvm_exit()`, in questo modo la routine che la rimpiazza (`wamm_exit()`) esegue tutte le operazioni necessarie per la conclusione del monitoraggio, compreso, quando necessario, l'invio del file di traccia prodotto dal task PVM.

- Ogni task affinché possa ricevere messaggi di *steering* deve utilizzare una funzione receive con *wild-card* (con parametri (-1, -1)), in questo

Funzioni Aggiuntive	Descrizione delle routine
wammOL_print	Permette la visualizzazione dei messaggi provenienti dai task oggetto del monitoraggio sulla finestra text_output dell'interfaccia.
wamm_close	Permette di chiudere una fase di monitoraggio, consentendo che i task continuino l'elaborazione senza più raccogliere dati di traccia.
wamm_open	Consente di aprire una nuova fase di monitoraggio nel caso in cui sia stata precedentemente eseguita una wamm_close.
wamm_print	Permette di rilevare eventi definiti dall'utente.
wamm_scandouble	Consente di effettuare operazioni di input verso un task, con valori di tipo double.
wamm_scanfloat	Consente di effettuare operazioni di input verso un task, con valori di tipo float.
wamm_scanint	Consente di effettuare operazioni di input verso un task, con valori di tipo int.
wamm_scanlong	Consente di effettuare operazioni di input verso un task, con valori di tipo long.
wamm_scanstring	Consente di effettuare operazioni di input verso un task, con valori di tipo string.
wamm_setsampling	Permette di cambiare la frequenza di campionamento.

Tabella 2: Elenco delle funzioni aggiunte alla libreria fornita da WAMM.

modo il task può ricevere un messagio con qualsiasi TAG da qualsiasi task.

Oltre a ciò, è necessario che l'applicazione includa lo header file `wammOL.h`. Questo deve essere messo immediatamente sotto la specifica dello header file di PVM (`pvm3.h`). Il file `wammOL.h` contiene le macro che sostituiscono le routine PVM con chiamate alle routine della libreria di WAMM (`wammtracelib.a`).
Per sfruttare alcune caratteristiche della libreria di WAMM, è necessario aggiungere al codice dell'applicazione l'invocazione esplicita alla routine desiderata.
Ad esempio, WAMM prevede che come periodo di campionamento possa essere utilizzato quello di default (5 secondi). Se l'utente volesse cambiare la frequenza di campionamento potrebbe procedere in più modi: inserendo immediatamente prima della `pvm_mytid()` l'invocazione alla routine `wamm_setsampling(n,t)`, in questo modo il campionamento verrà effettuato ogni `n` invocazioni a funzioni PVM, se `n` è maggiore di uno, oppure ogni `t` secondi, altrimenti; alternativamente l'utente, se ha utilizzato all'interno della sua applicazione delle *receive* con *wild-card*, può agire a tempo di esecuzione su uno *slider* posto nella finestra di monitoring usata dall'interfaccia.
L'invocazione alle routine `wamm_open()` e `wamm_close()` permette, invece, di

monitorare solo alcune fasi dell'esecuzione dell'applicazione. Questa caratteristica è senz'altro molto interessante perchè nel caso in cui si è interessati solo ad osservare solo piccole frazioni dell'esecuzione, evita che vengano generate grosse quantità di informazioni.
— Altre routine utili sono le `wamm_scan*()` (dove l'asterisco può essere sostituito da `int`, `string`, ecc.) che possono essere aggiunte nel programma per poter effettuare input di differente tipo; in tal caso, durante l'esecuzione verrà aperta un'apposita finestra, come mostrato in Figura 4, indicante il tipo di input richiesto ed il TID del task richiedente.
La routine `wamm_printOL()`, invece, che ridirige l'output del programma ver-

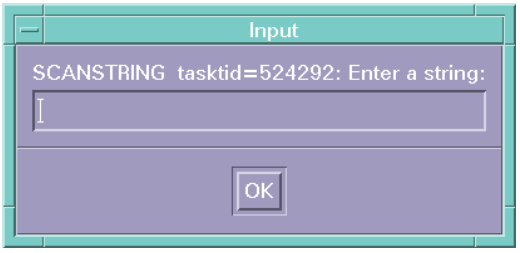

Figura 4: Finestra di controllo per eseguire operazioni di input verso i processi

so l'interfaccia, può essere utilizzata quando non si è interessati ad analizzare tutto l'output prodotto da un task (cosa ottenibile tramite WAMM, mediante l'attivazione dell'opzione **Output**), ma solo un suo sottoinsieme.

Un esempio di uso delle routine instrumentate è mostato in Figura 5.

Infine, mediante l'utilizzo della routine `wamm_print()` (i cui parametri seguono una formattazione simile a quella della `printf()` del C) è possibile rilevare eventi definiti dall'utente.
Una volta effettuate queste modifiche, il codice può essere compilato e per la generazione del modulo eseguibile deve essere utilizzata sia la libreria PVM `libpvm3.a` che la libreria instrumentata di WAMM `wammtracelib.a`. Scrivendo in modo opportuno il `Makefile` (assieme al codice di WAMM, viene fornito all'interno della directory `example`, il codice sorgente relativo ad un programma esempio ed il `Makefile` da utilizzare per la compilazione di applicazioni instrumentate) è possibile utilizzare WAMM per eseguire la compilazione remota del programma su tutti i nodi del metacomputer.

```
Task A: master
.......
#include "wammOL.h"
.......
pvm_mytid();
pvm_spawn ("B",...,nproc,tid);
.......
.......
wamm_print("identificatore evento")
.......
.......
pvm_exit;
------------------------------------------------------------
Task B: master
.......
#include "wammOL.h"
.......
pvm_mytid();
wamm_close();
.......
.......
wamm_open();
.......
.......
pvm_exit;
```

Figura 5: Esempio di uso di routine PVM instrumentate.

5.1.5 Attivazione e controllo di processi

Per eseguire un programma PVM è possibile utilizzare la finestra di dialogo richiamabile per mezzo della voce `Spawn` del menu `Apps` (Fig. 6). Il dialog box permette di specificare i parametri di attivazione necessari:

- **Name**: nome del programma da eseguire;
- **Arguments**: argomenti da passare al programma;
- **Copies**: numero di copie del programma di cui si vuole attivare l'esecuzione;
- **Where**: permette di scegliere dove eseguire le varie copie del programma. Le scelte possibili sono:
  - **Auto**: gli host su cui eseguire le varie copie del programma vengono scelti in maniera automatica da PVM, questa opzione corrisponde al flag `PvmTaskDefault` di PVM;
  - **Arch**: è necessario specificare una architettura su cui eseguire i processi. PVM sceglie tra le macchine aventi l'architettura indicata. Se viene attivato il *checkbutton* **!=** (corrispondente al flag PVM `PvmHostCompl`), PVM sceglie una architettura diversa da quella specificata. La voce **Arch** corrisponde al flag `PvmTaskArch` di PVM;
  - **Host**: è necessario specificare il nodo su cui eseguire i processi. Anche in questo caso se viene selezionato **!=**, PVM sceglie un host diverso da quello specificato. La voce **Host** corrisponde al flag `PvmTaskHost` di PVM;
- **Output**: questo parametro permette di ridirigere l'output dei processi eseguiti verso l'interfaccia. In questo modo è possibile avere l'output dei vari processi in finestre separate. Questa funzionalità viene implementata con la modifica del valore della variabile `PvmOutputTid` di PVM;
- **Debug**: permette di eseguire il processo specificato sotto il controllo di un debugger. Corrisponde al flag `PvmTaskDebug` di PVM;
- **MPP**: avvia i processi sul *front-end* di una macchina multiprocessore anziché sui suoi nodi. Corrisponde al flag `PvmMppFront` di PVM;

- **Sampling**: in questo modo, i processi che vengono attivati, spediranno periodicamente all'interfaccia informazioni descriventi la propria esecuzione, mentre i processi `PVMTasker`, invieranno all'interfaccia le informazioni riguardanti il carico sulle macchine;

- **Tracing**: in questo modo i processi vengono avvertiti che dovranno effettuare la rilevazione degli eventi;

- **Tracefile name**: permette all'utente di scegliere un nome per il file di traccia che verrà generato. Il nome di default è `wammfile.uid.n.anim`, dove `uid` indica il numero identificativo dell'utente, `n` serve a distinguere i vari file di traccia relativi a differenti fasi della stessa esecuzione e `anim` contraddistingue i file di traccia che possono essere visualizzati con lo strumento `PVaniM` [TSS96a]. Il file di traccia così generato verrà collocato, per default, nella directory `/temp`. Tramite il popup menu è possibile selezionare la voce **filename**, in quel caso apparirà una finestra che consentirà di scegliere la directory che dovrà contenere il file di traccia e di specificare il prefisso che si vuole assegnare al file generato.

Una volta impostati tutti i parametri, selezionando il bottone **Spawn!**, viene attivata l'esecuzione del programma specificato. Per ogni copia del proces-

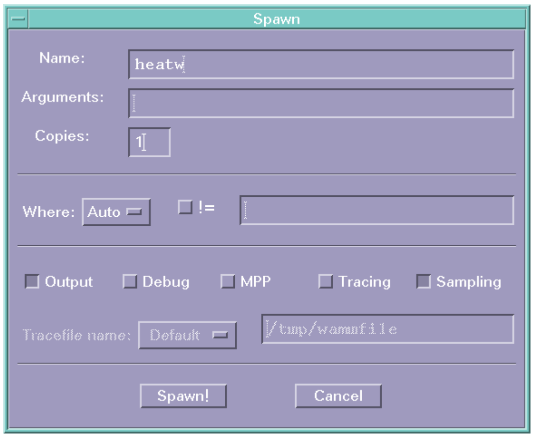

Figura 6: Finestra di controllo per lo Spawn

so avviata con successo verranno mostrati, nell'area messaggi della finestra WAN:

- il nome del processo;

- il TID;

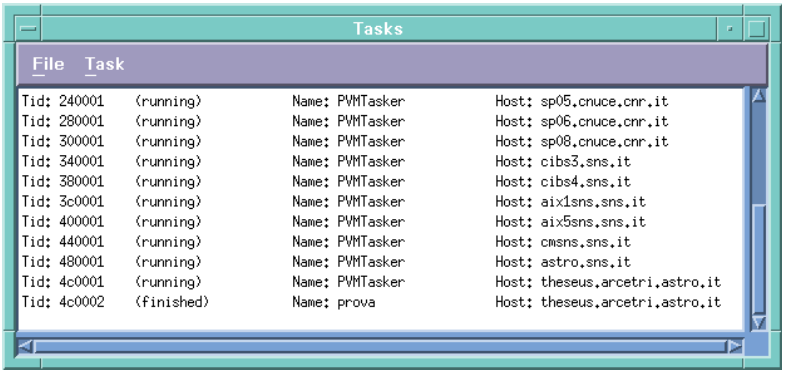

Figura 7: Finestra di controllo per i tasks

- il nome del nodo su cui è stato mandato in esecuzione.

In caso di fallimento, verrà riportato un messaggio di errore.

I task PVM in esecuzione sulla macchina virtuale possono essere controllati utilizzando la finestra mostrata in Figura 7, richiamabile con il comando `Tasks` del menu `Windows`. La finestra contiene una lista dei processi PVM. Per ogni processo vengono mostrati: il TID, lo stato (**running** oppure **finished**), il nome e l'indirizzo della macchina su cui viene eseguito. Per default, la lista dei processi non comprende i task di servizio (`PVMTasker`, `PVMHoster` e `PVMMaker`); possono essere mostrati/nascosti cambiando lo stato della voce `Show system tasks` del menu `File`. Lo stato dei task viene aggiornato automaticamente da `wamm` sulla base delle informazioni provenienti dai processi `PVMTasker` in esecuzione sui nodi della macchina virtuale. Per rimuovere le righe relative a task già terminati occorre richiamare `Remove old tasks` nello stesso menu.

Agendo con un doppio click su una riga, è possibile aprire una finestra che ne mostra l'output prodotto dal corrispondente processo; questa operazione può essere anche svolta su un gruppo di processi, selezionando le righe corrispondenti e richiamando il comando `Output` del menu `Task`. Questa funzionalità è disponibile solo per i processi attivati con l'opzione **Output** (cfr. fig. 6).

La finestra consente anche di terminare uno o più processi: dopo aver selezionato le righe corrispondenti ai processi interessati, deve essere eseguito il comando `Kill` del menu `Task`. In maniera analoga è possibile inviare segnali ai processi selezionati (`Signal`, menu `Task`).

Una volta attivata l'esecuzione dell'applicazione, nel caso in cui sia stata impostata l'opzione **Tracing** e non quella **Sampling**, al termine dell'esecuzione dell'applicazione, verrà effettuata, in automatico, la raccolta dei file di traccia prodotti e verrà eseguita la fase di elaborazione del file per effettuare l'ordinamento dei dati di traccia; anche in questo caso l'utente verrà informato sull'esito delle operazioni.

Se l'applicazione, invece, viene attivata con l'opzione **Sampling**, apparirà una nuova finestra **Monitor**.

5.1.6 Chiusura dell'interfaccia

La chiusura del programma può avvenire in due modalità differenti, richiamabili dal menu `File` della finestra WAN:

Quit: esce dal programma, chiedendo conferma. I processi di servizio vengono terminati (`PVMTasker`, `PVMHoster`, `PVMMaker`);

Halt: esce dal programma richiamando `pvm_halt()`. La macchina virtuale viene distrutta e tutti i processi PVM in esecuzione vengono terminati. Viene chiesta conferma.

I processi di servizio controllano periodicamente lo stato del processo `wamm`: nel caso di chiusura anomala dell'interfaccia, tali processi terminano autonomamente.

5.2 Finestra Monitor

Nella finestra Monitor vengono mostrate in forma grafica tutte le informazioni raccolte periodicamente durante l'esecuzione di una applicazione PVM.
Come mostrato in Figura 8, la finestra che visualizza le informazioni relative al monitoraggio è divisa in 3 parti: nel riquadro in alto a sinistra, vengono mostrate le varie viste grafiche riportanti informazioni sulle macchine e sui task PVM; a destra, sono elencati i processi sottoposti al monitoraggio ed in basso, vengono mostrati i messaggi relativi sia ai processi che alle macchine monitorate. La parte superiore della vista grafica, mostra le informazioni riguardanti i vari host che compongono la macchina virtuale:

- *Host List*: questa vista illustra, in modo testuale, gli host che compongono la macchina virtuale ed i task allocati su ciascuno di essi. I task sono indicati con un valore intero.

- *Load Info*: questa vista fornisce informazioni sul carico di ogni host mostrando il numero medio di job nella coda di esecuzione.

- *Host Utilization*: in tale vista viene mostrata la percentuale di tempo che ogni processore trascorre in ognuno dei tre stati (computazione,

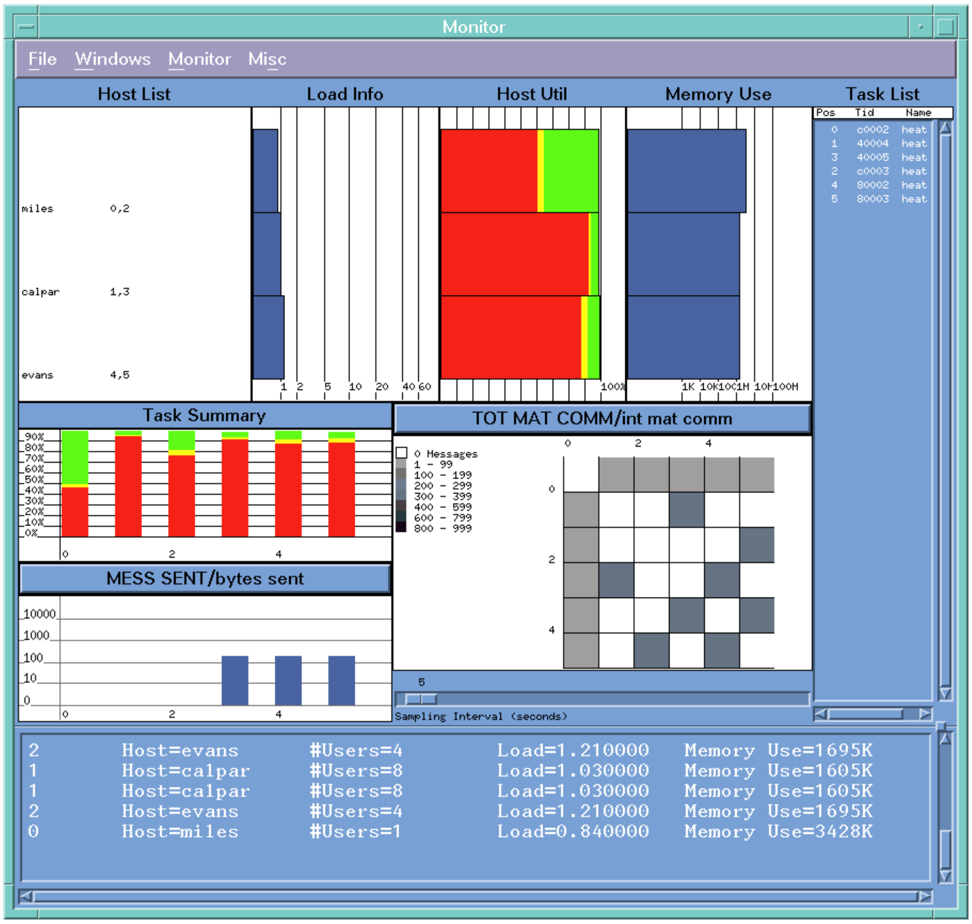

Figura 8: Finestra per la visualizzazione dei dati di trace prodotti durante l'esecuzione di un'applicazione PVM.

spedizione e ricezione di messaggi). Per far questo, ogni processore è associato ad una barra orizzontale suddivisa in segmenti colorati in accordo allo stato in cui esso si trova per l'intervallo di tempo specificato dalla loro lunghezza. I colori verde, giallo e rosso indicano, rispettivamente, computazione, invio di un messaggio e attesa per la ricezione di un messaggio.
La vista Host Utilizzation è analoga alla vista Gantt fornita da ParaGraph, va sottolineato che il livello di dettaglio è sicuramente inferiore, ma ha il pregio di fornire, per ogni processore, una composizione delle informazioni provenienti dall'esecuzione di tutti i task su di esso allocati.

- *Memory Use*: questa vista illustra, per ogni host, l'ammontare di memoria utilizzata dai task PVM su di esso allocati.

Le viste precedenti sono "cliccabili", in questo modo l'utente può ottenere informazioni testuali riguardanti il carico dell'host selezionato, che appariranno in basso nella finestra **Monitor**.
La parte inferiore della vista grafica mostra invece informazioni riguardanti i task PVM e la comunicazione da essi effettuata:

- *Task Summary*: questa vista presenta lo stesso tipo di informazioni fornite dalla vista Host Utilization, ma in questo caso tali informazioni sono relative ai singoli task. Confrontando le informazioni fornite dalla vista *Task Summary* con quelle fornite dalla vista *Host Utilization*, l'utente può comprendere qual'è il task maggiormente responsabile di eventuali degradi delle prestazioni;

- *MESS SENT/bytes sent*: questa vista fornisce informazioni riguardanti sia il numero dei messaggi, che il numero di byte spediti da ogni task. Tali informazioni si riferiscono esclusivamente alla comunicazione avvenuta durante il periodo di campionamento, ma in ogni caso fornisce una idea della comunicazione e permette di individuare dei comportamenti anomali;

- *TOT MAT COMM/int mat comm*: questa vista è rappresentata da una matrice le cui righe e colonne corrispondono rispettivamente ai processi mittente e destinatario. La casella corrispondente ai processori coinvolti nella comunicazione viene colorata in accordo al numero dei messaggi scambiati. Le informazioni fornite possono riguardare le comunicazioni relative al periodo di campionamento, o all'intera esecuzione. Questa vista, se viene impostata per controllare le comunicazioni nel periodo di

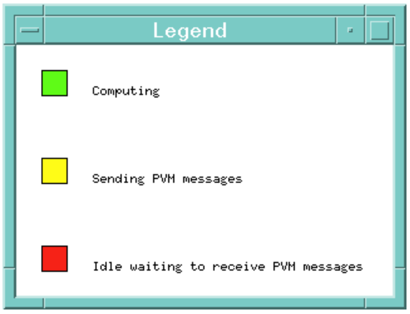

Figura 9: La vista Legend della finestra Monitor.

campionamento, è “cliccabile” e ciò permette di cambiare l'associazione tra i diversi colori ed il numero dei messaggi che essi rappresentano.

Utilizzando il comando `Legend` del menu `Monitor` è richiamabile una finestra che mostra l'associazione tra i colori utilizzati nelle varie viste e gli stati assunti dai processi e dai processori (vedi Figura 9).

Oltre alle viste sopra descritte, nella finestra **Monitor**, è presente anche uno *slider* che permette all'utente di cambiare la frequenza di campionamento. In questo modo, l'utente può decidere il grado di intrusione dello strumento e il livello di dettaglio delle viste risultanti.

Come si puó osservare in Figura 8, nella *Task List* (riquadro a destra della finestra **Monitor**) vengono elencati i task PVM che partecipano al monitoraggio. Per ogni task vengono presentate le seguenti informazioni: il numero intero che lo identifica, il TID ed il nome del programma. Ogni elemento della lista è “cliccabile” e mostra, in una finestra separata, l'output del task selezionato.

Nel riquadro in basso della finestra, invece, vengono mostrate le informazioni riguardanti l'ingresso e l'uscita dal monitoraggio dei task PVM, le informazioni sul carico degli host ed i vari messaggi di output prodotti dai task PVM, nel caso in cui siano state utilizzate routine `wamm_printOL()`.

Il comando `Converter` del menu `Monitor` consente di attivare una finestra di selezione dei file di traccia in formato leggibile da pvanim, in modo che questi possano essere convertiti nel formato leggibile dal tool ParaGraph.