



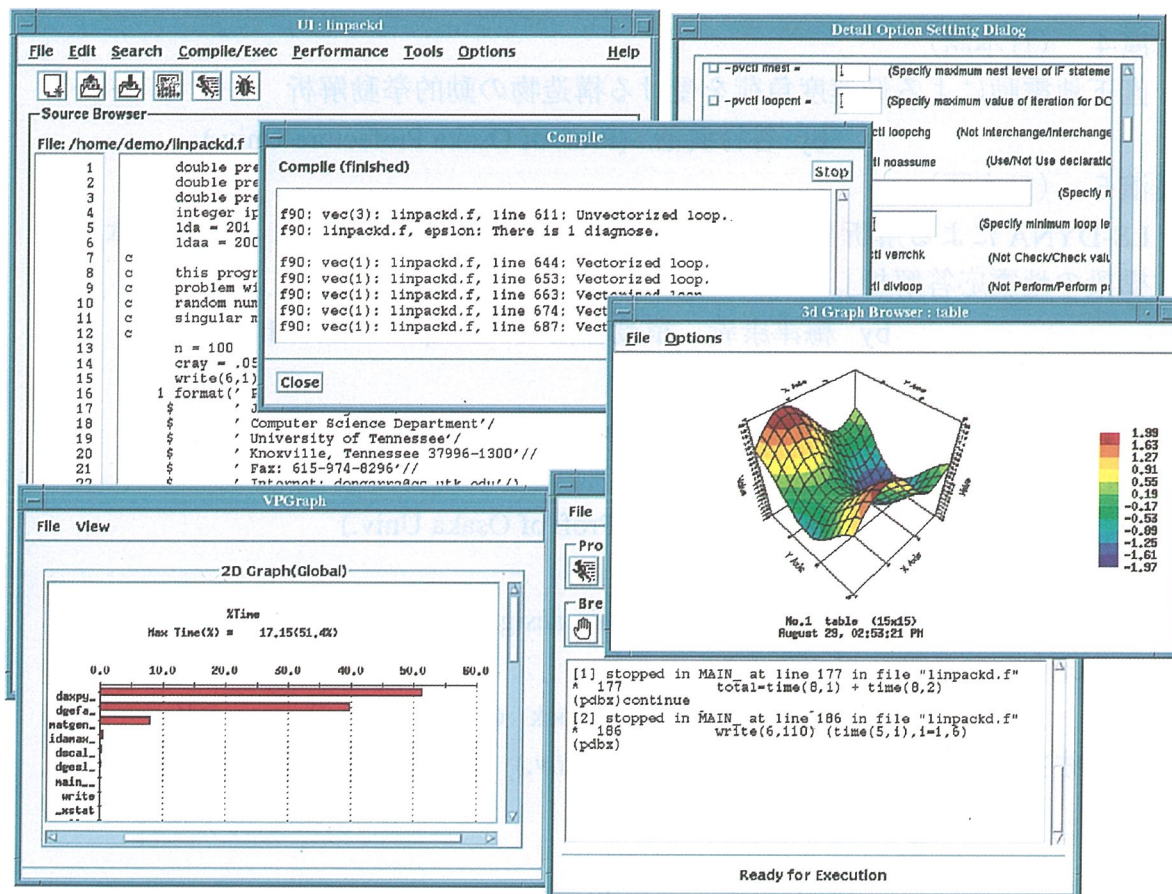
Title	センターだより 大阪大学大型計算機センターニュース 第111号 (Vol.28 No.4)
Author(s)	
Citation	大阪大学大型計算機センターニュース. 1999, 111, p. 24-38
Version Type	VoR
URL	<a href="https://hdl.handle.net/11094/66319">https://hdl.handle.net/11094/66319</a>
rights	
Note	

*The University of Osaka Institutional Knowledge Archive : OUKA*

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

## 統合プログラム開発環境PSUITE概要



1. はじめに
  - ツールの要件
2. PSUITEの特徴
  - PSUITEの制限事項
  - PSUITEのシステム環境
  - PSUITEの構成
3. プログラム開発支援
4. 最適化ブラウザ
5. デバッガ
6. 性能解析
7. プロジェクト管理
8. その他

## 1. はじめに

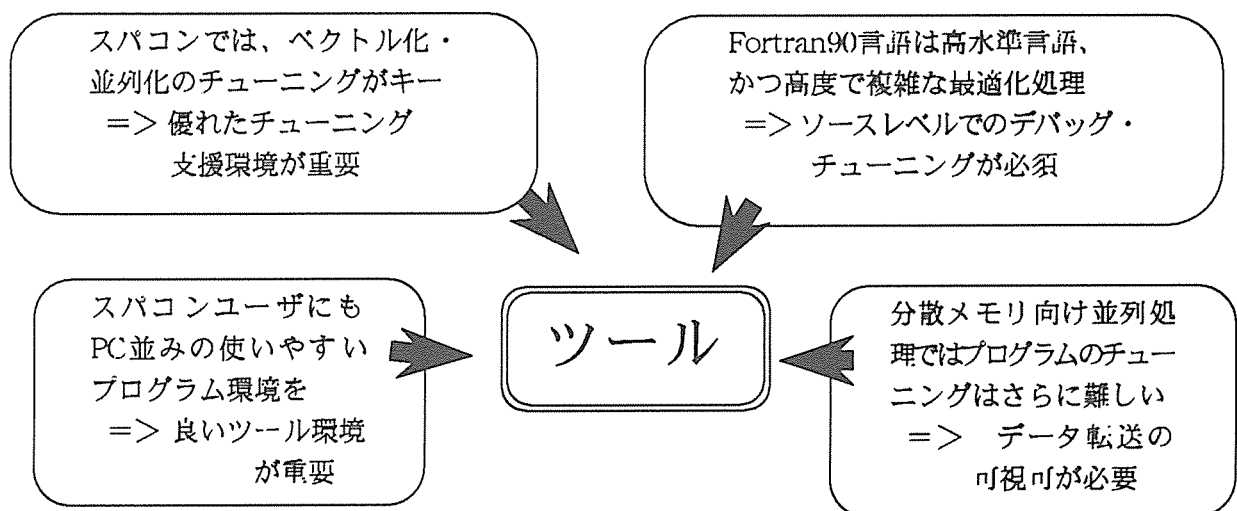
スーパーコンピュータ（以下スパコンと略す）の世界では、ハードウェア性能をいかにしてかつ容易に引き出すかが鍵であり、そのためには最適化・ベクトル化・並列化を推進する性能向上支援ツールが重要な位置を占めてきている。

一方、利用者にとって見れば、今やスパコンといえども特別なマシンではなく、GUIベースの使い勝手の良い開発環境が必要となっている。

さらに、最近の動向として、手続きを呼び出し元に展開する「手続きのインライン展開」や複数のループを一つにまとめる「ループ融合」等の高度な最適化、あるいはFortran90で追加された「配列構文」等により、元のソースプログラムと実際のコードの対応が複雑になっており、ソースプログラムレベルでのデバッグ・チューニングが困難になっているが、利用者から見れば、あくまで元のソースプログラムレベルで行えることが望まれている。

ここで紹介する統合プログラム開発環境PSUITEは、これらの要求に応えるべく開発された製品である。

### ツールの要件



## 2. PSUITEの特徴

- 統一されたGUIのもとで各種ツールを統合  
プログラムの開発サイクル(編集・翻訳・実行・デバッグ・チューニング)をサポート
- SXクロス環境(NEC,SUN,SGI,HP)
- コンパイラおよび各種ツール間連携による使い易い環境
- 高度に最適化されたプログラムに対してソースレベルのデバッグ・チューニングが可能
- FORTRAN90、Cに加え分散メモリ向け並列処理にも対応

## PSUITEの制限事項

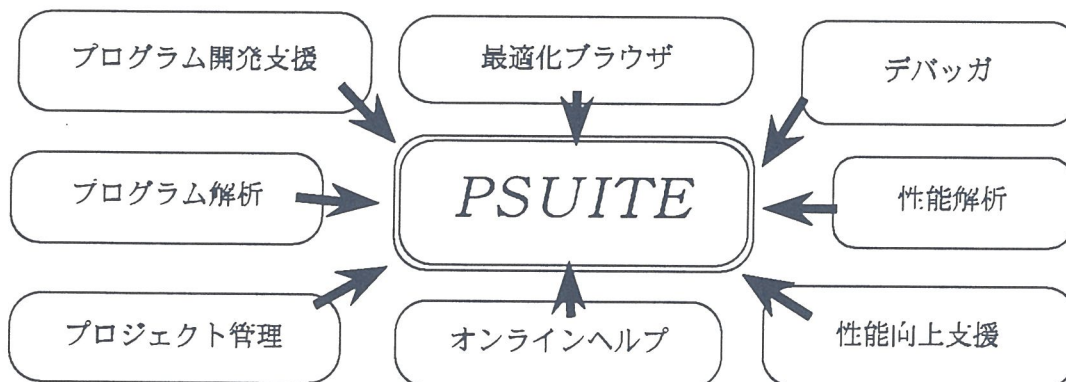
- × FORTRAN77の利用はできない。  
FORTRAN90への変換は可能。
- × 会話形式の資源の利用になる。  
NQSの利用が直接できない。  
MPIの利用はできない。(会話形式 = 1CPUのため)

PSUITEでは、これらの機能を下図のように、スパコンの負荷軽減、ワークステーションとスパコン間でのプログラム共用、スパコン利用時間に束縛されない利用環境等の目的で、プログラム実行以外はクロスコンパイラと一体となってワークステーション上で処理することにより、実現している。

## PSUITEのシステム環境



## PSUITEの構成

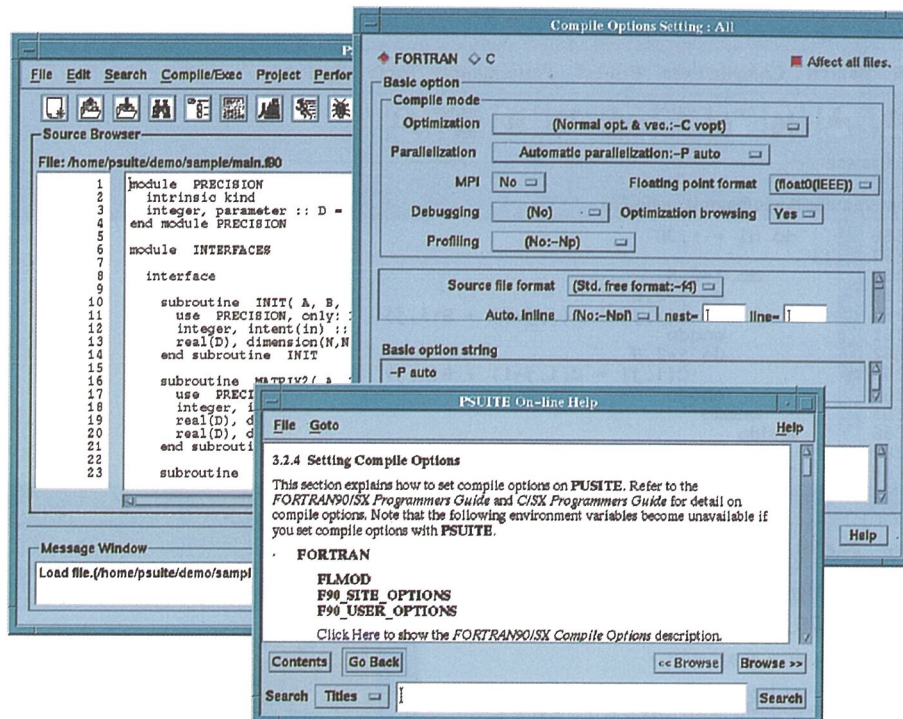




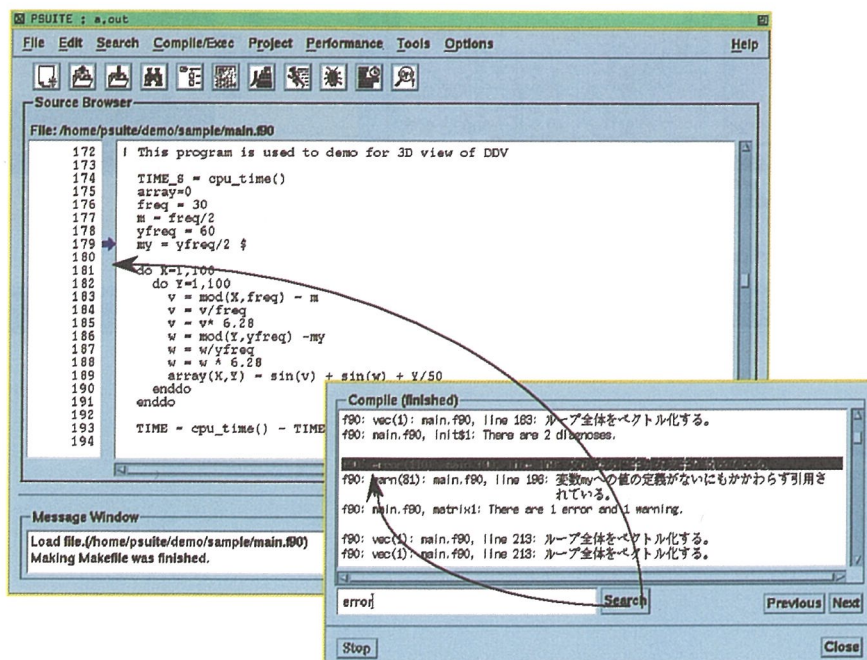
### 3. プログラム開発支援

- 各種編集機能つきソースブラウザ  
(edit,grep,find,replace,etc)
- クロスコンパコイラ、各種ツールの呼び出し、プログラム実行
- MakeFileの自動生成
- コンパイルエラーメッセージとソースプログラムの自動関連付け表示
- GUIによるコンパイラオプション設定

#### < 翻訳オプション設定 & HELP画面例 >



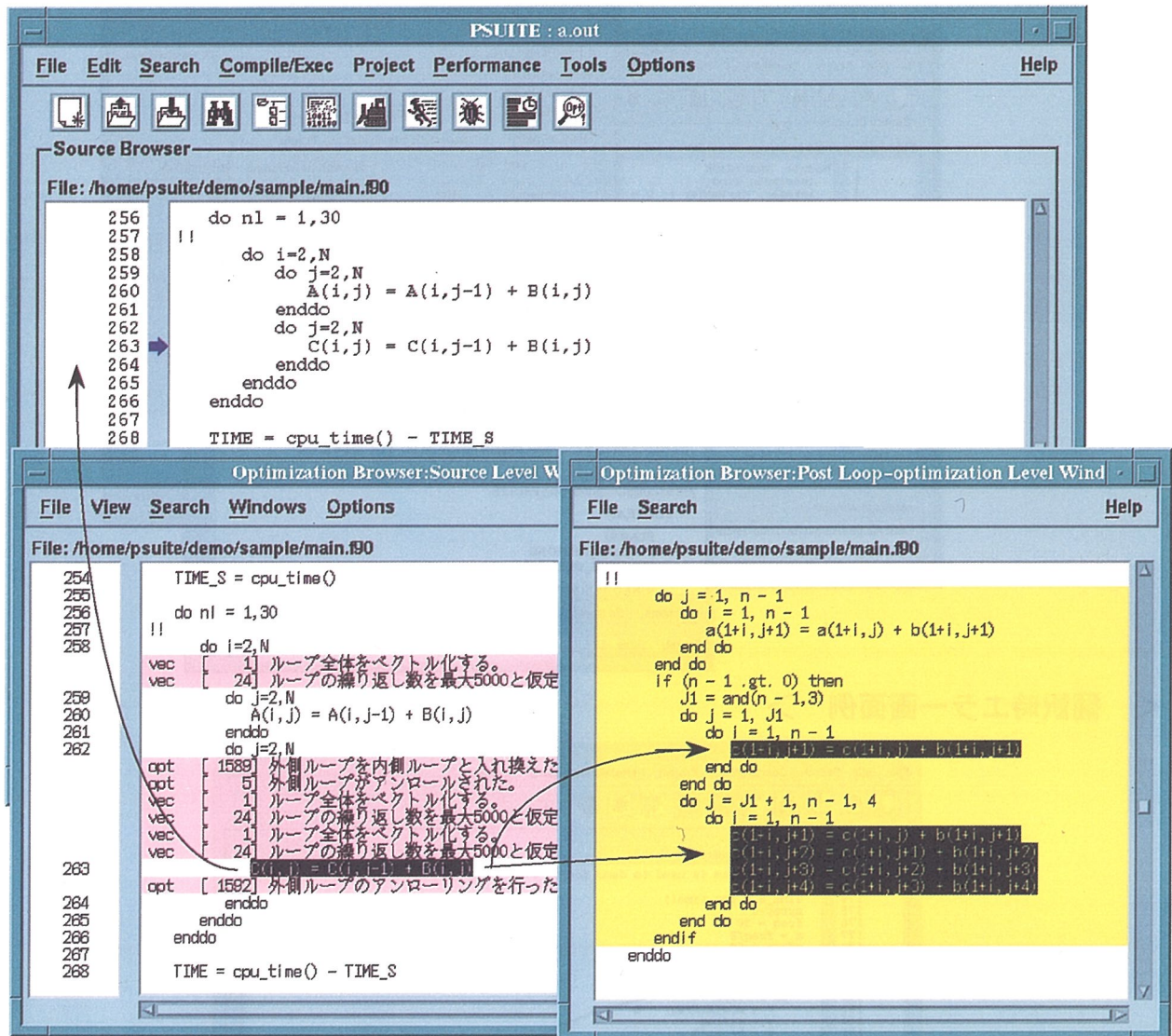
#### < 翻訳時エラー画面例 >



## 4. 最適化ブラウザ

- ・コンパイラの最適化・ベクトル化・並列化処理状況をソースプログラムと同じレベルで表示（配列構文、ループ変型、インライン展開、ベクトル化／並列化状況）
- ・文をクリックすると、ソースプログラムと同じレベルで対応する情報を表示

< 最適化ブラウザ画面例 >





## 5. デバッガ

- 簡単なコマンド操作

アイコンによるコマンド指定

ソースプログラム上でブレークポイントの設定や表示、データの選択が可能

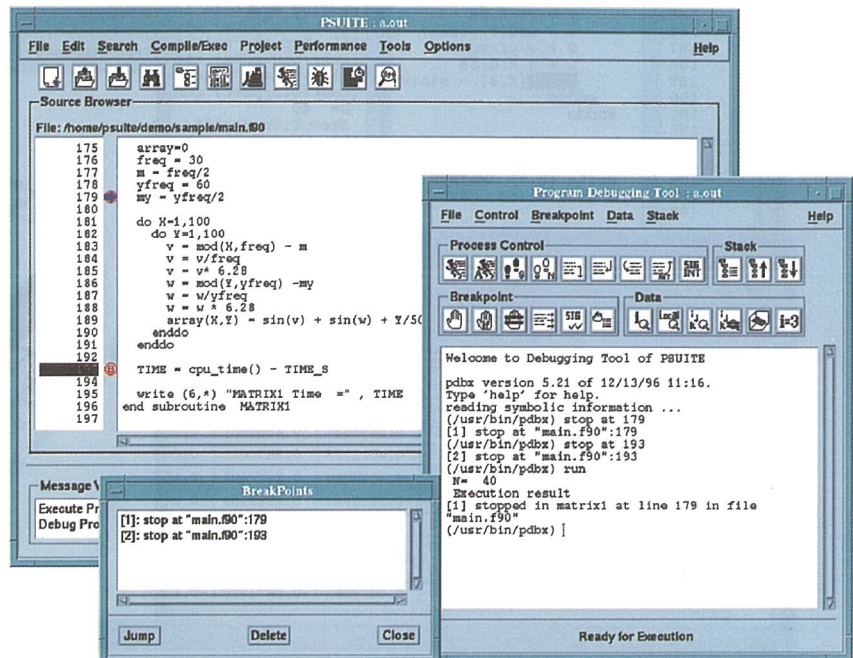
- ソースブラウザとの連携により対応するソースプログラムを表示

プログラム停止時

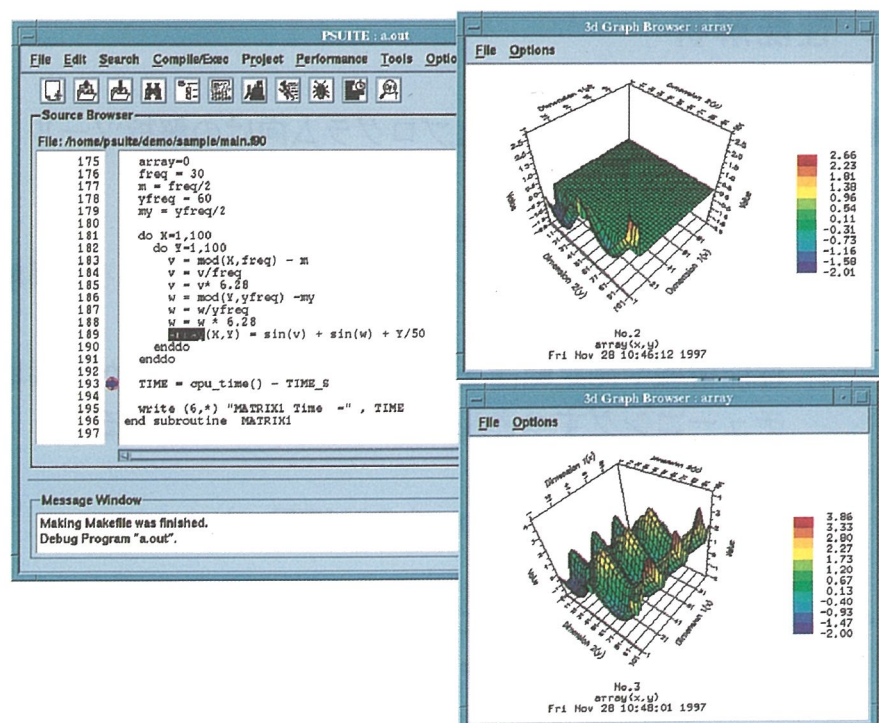
ブレークポイントやスタックトレース情報の表示指定時

- 配列データの2次示3次元グラフによる表示

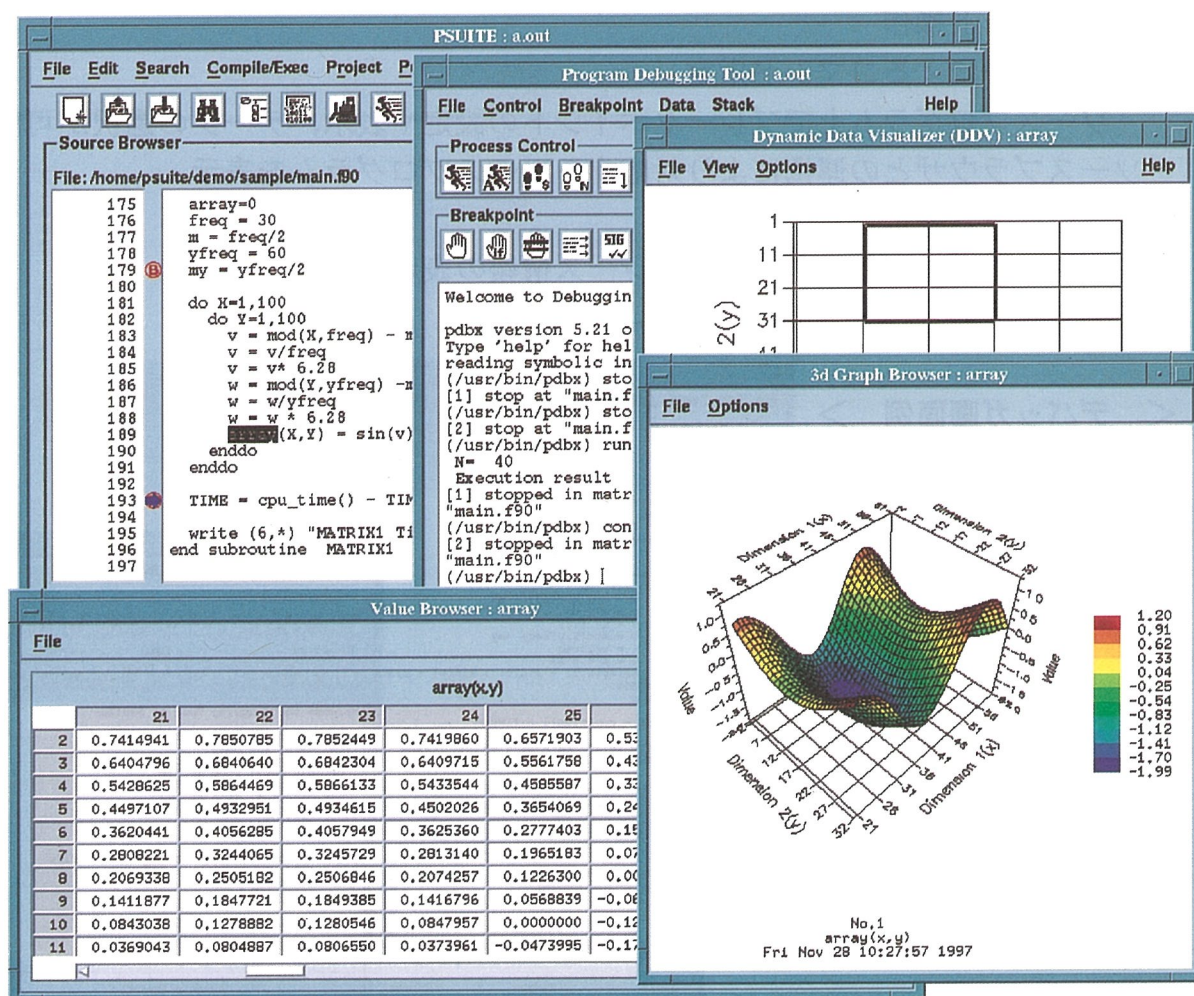
### < デバッガ画面例 >



### < デバッガによるデータの可視化 >



## ＜ デバッガによるデータの部分表示 ＞



## 6. 性能解析

### ・最適化・ベクトル化・並列化プログラム向けの解析ツール

#### Vprof

情報が簡単に採取可能

チューニングすべきルーチンの検出に適する（初期解析向け）

#### PSUITEpa

性能情報や測定範囲を選択して、必要な情報だけ採取可能

チューニングすべきループの絞り込みに適する

### ・性能情報の可視化

2次元・3次元グラフ(Vprof, PSUITEpa)

Callグラフ(PSUITEpa)

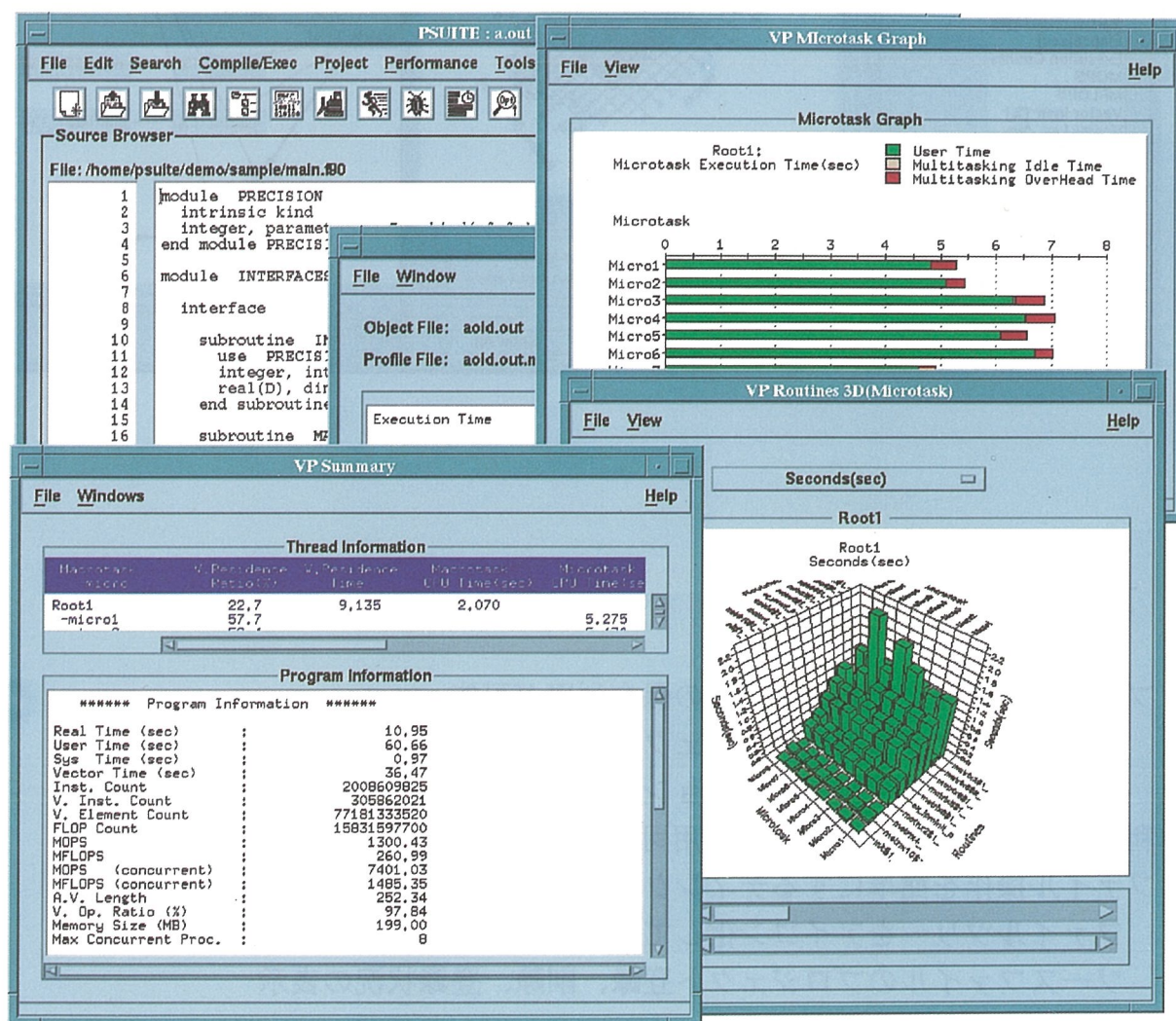
### ・ソースブラウザとの連携により対応するソースプログラムを表示

グラフ上の性能情報をクリックした時(PSUITEpa)



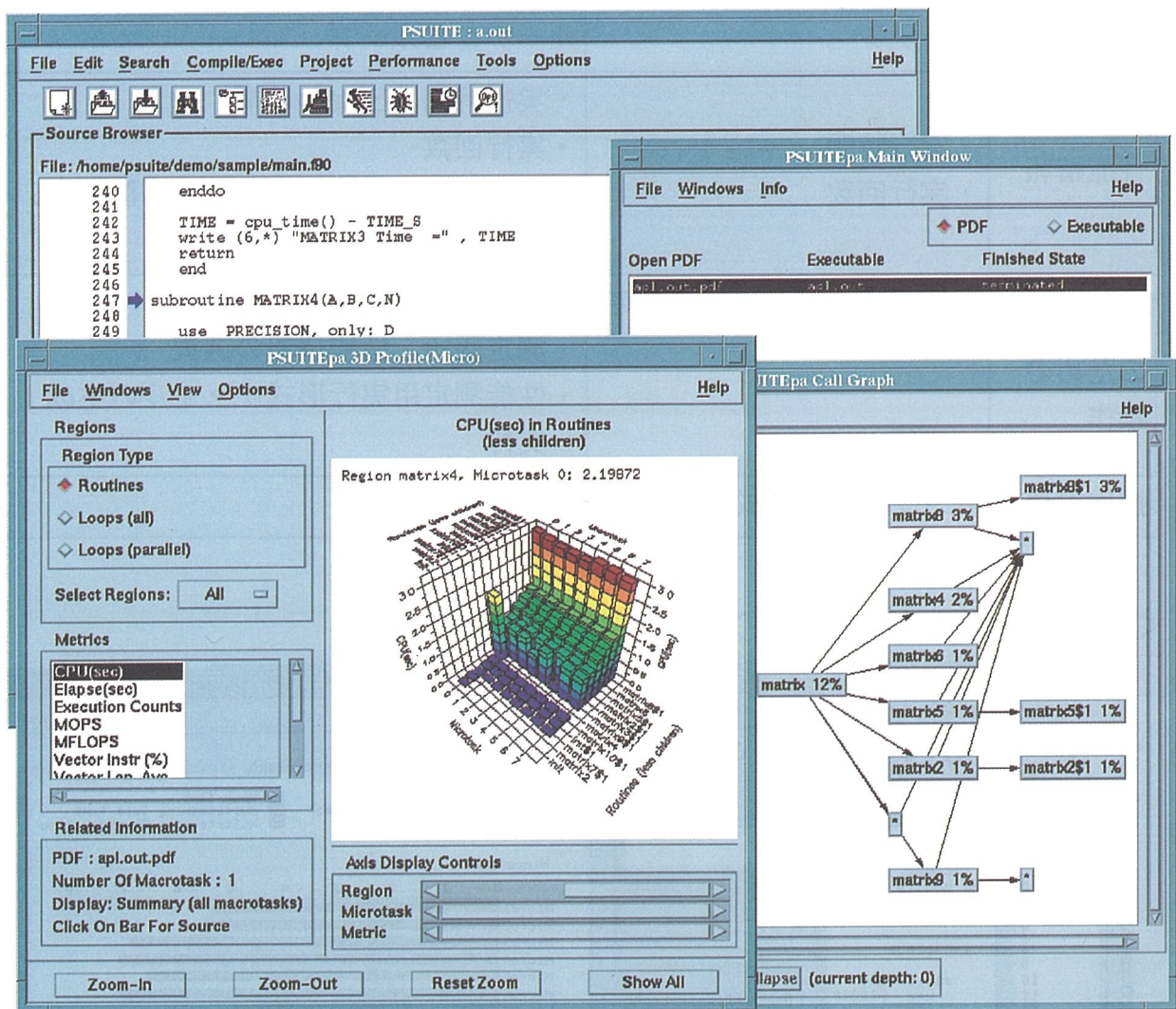
	<i>Vprof</i>	<i>PSUITEpa</i>
測定区間	ルーチン	ルーチンとループ
性能情報	<ul style="list-style-type: none"> <li>・ 実行時間</li> <li>・ 実行回数</li> </ul>	<ul style="list-style-type: none"> <li>・ 実行時間、Elapse時間</li> <li>・ 実行回数</li> <li>・ ハードウェア情報（ベクトル演算率、ベクトル長、MFLOPS等）</li> </ul>
情報採取のための操作	<ul style="list-style-type: none"> <li>・ 翻訳時オプションの指定</li> <li>・ 実行</li> </ul>	<ul style="list-style-type: none"> <li>・ 翻訳時オプションの指定</li> <li>・ 測定箇所、性能情報の選択</li> <li>・ 性能測定用実行形式ファイルの生成</li> <li>・ 実行</li> </ul>
採取方法	サンプリング方式	実測方式

### < 性能解析(Vprof)画面例 >





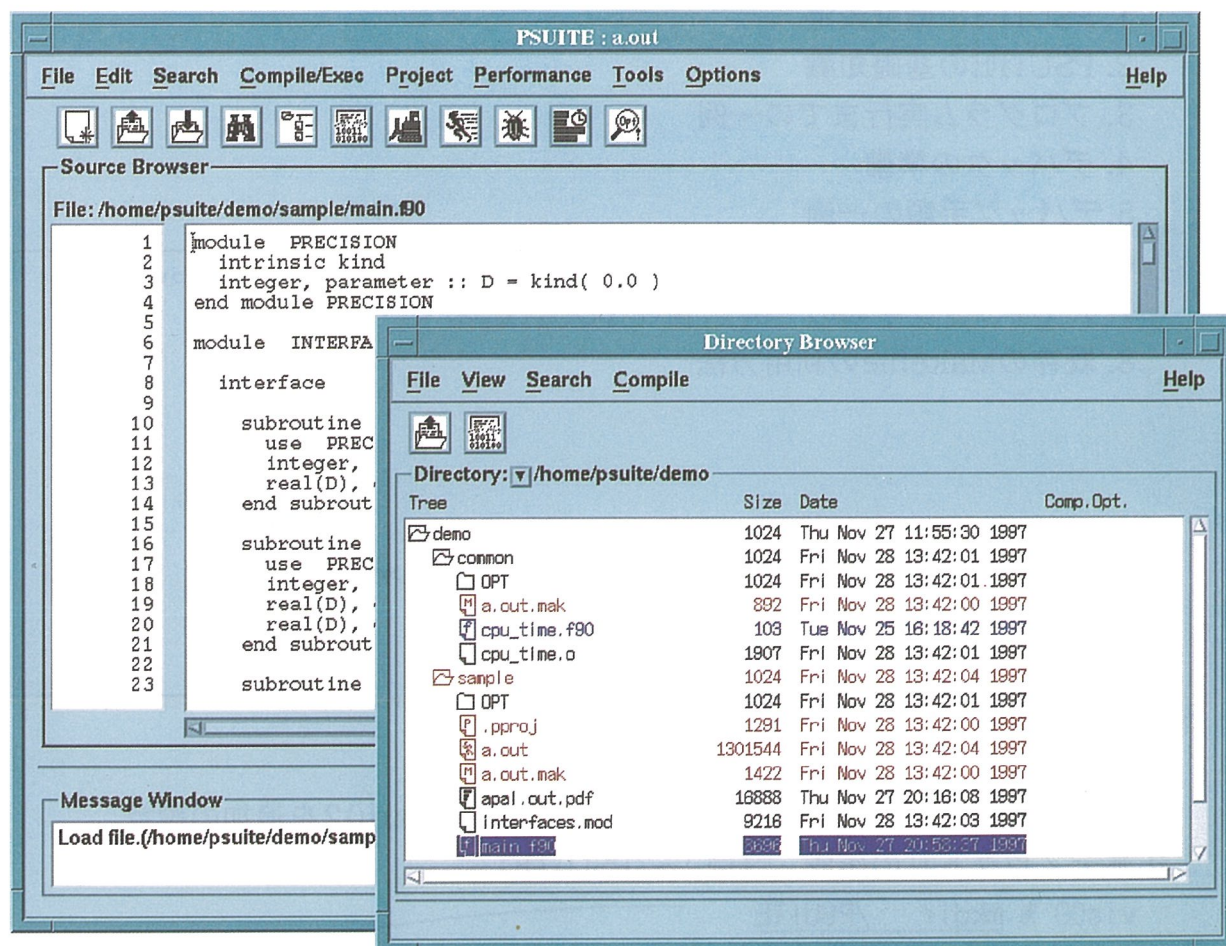
## < 性能解析(Psuitepa)画面例 >



## 7. プロジェクト管理

- ・プログラム開発上で必要な情報の保存利用が可能  
ソースプログラム毎にプロジェクトとして一括管理  
ソースファイル、オプション、Makefile、実行情報、等
- ・利用者が作成したMakefileを利用可能
- ・ファイル操作を簡単にするディレクトリブラウザ  
ファイルツリーをアイコン化して表示  
ソースファイルのプロジェクト登録、削除、登録状況の表示  
翻訳時オプションの表示  
ファイル名をクリックすると対応するソースファイルを表示

## < プロジェクト画面例 >



## 8. その他

### ・ オンラインヘルプ機能

PSUITEの機能、操作方法

コンパイラの翻訳時オプション、実行時オプション

- ・ FORTRAN77言語で記述されたソースプログラムをFORTRAN90言語機能を利用したソースプログラムに変換する機能



## PSUITEを簡単に御利用いただくために

1. PSUITEの環境設定
2. PSUITEの基礎知識
3. プログラム実行までの一例
4. デバックの準備
5. デバック手順の一例
6. チューニングの準備 (Vprof,PSUITEpa)
7. チューニング手順の一例 (Vprof,PSUITEpa)
8. 既存のMakefileの利用方法

```
% cat .rhosts
vls00
vls01
vls02
vls03
vls04
vls05
vls06
vls07
vls08
vls09
vls10
visd01
visd02
```

### 1. PSUITEの環境設定

SX4における設定

SX4とのリモート接続環境

~/rhostsファイルにvis01~vis10,visd01,visd02を追加の設定

作業ディレクトリの作成 (例: ~/PSUITE)

vis09 % mkdir ~/PSUITE

WS(vis01~vis10,visd01,visd02)における設定

環境変数PATH--> PSUITEインストールディレクトリの設定

vis09 % set path=(\$path /usr/psuite)

(センター提供の環境ファイルを利用している方は不要)

環境変数DISPLAY--> X環境の設定

vis09 % setenv DISPLAY "unix:0.0"

.psuiteファイル--> PSUITE初期化ファイルの設定

vis09 % cp /usr/psuite/.psuite ~/.

エディタで.psuiteファイル中の以下の項目に値を設定

PSUITE\*remotehost: sx4.center.osaka-u.ac.jp (sxのホスト名)

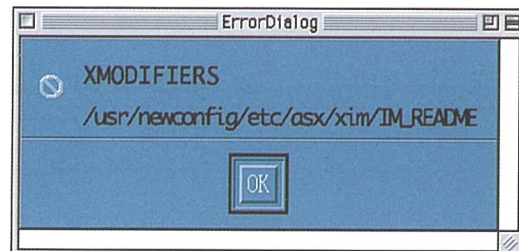
PSUITE\*remotedir: ~/PSUITE (作業ディレクトリ (例))

PSUITE\*username: 利用者番号 (利用者番号)

実行コマンド

vis09 % psuite

(日本語環境が整っていない場合はError Dialogが表示されるが、無視する。)



## 2. PSUITEの基礎知識

### 操作の基本

- メニューやアイコンをポイント --> クリック
- ソースブラウザがPSUITEの中心ツール
- 各種オプションや情報の設定
- プログラムのコンパイル・実行形式ファイルの作成・実行の指示
- コンパイラや各種ツールの起動
- プロジェクトは自動生成・自動更新
- 通常は、特別に意識する必要はない
- PSUITEが起動された作業ディレクトリ配下にプロジェクトファイル(.pproj)を自動作成
- 作業ディレクトリ配下のソースファイルを自動登録
- 各種オプション、実行に関する情報等も自動登録
- ディレクトリブラウザでプロジェクトへの登録状況の表示や登録・削除が可能

## 3. プログラム実行までの一例

### .1 ソースプログラムのオープン

メニューバーのFileからOpenFile...を選択し、目的のソースプログラムをオープンする

**File Edit Search Compile/Exec Project Performance Tools Options**

### .2 コンパイルオプションの設定

メニューバーのCompile/ExecからSetCompileOptions...を選択し、現れたウィンドウ上で必要なコンパイルオプションを指定

### .3 実行形式ファイルの生成

Buildアイコン  の選択... Makefileの自動作成、コンパイル、リンク

### .4 プログラム実行情報の設定

メニューバーのCompile/ExecからSetRunOptions...を選択し、現れたウィンドウ上で、引数、入力ファイル名・出力ファイル名、環境変数などを指定

### .5 プログラムの実行

Runアイコン  の選択

## 4. デバックの準備

### .1 デバッグ用コンパイルオプションの設定

メニューバーのCompile/ExecからSetCompileOptions.../ThisFile...を選択し、現れたウィンドウ上でDebuggingオプション (YES) を指定


### .2 実行形式ファイルの生成

Buildアイコンの選択... Makefileの自動作成、コンパイル、リンク

.3 プログラム実行情報の設定

メニューバーのCompile/ExecからSetRunOptions...を選択し、現れたウィンドウ上で、実行時に必要な情報を指定

.4 デバッガの起動


Debugアイコンの選択

## 5. デバッグ手順の一例

.1 ブレークポイントの設定

ソースブラウザ中の行を選択後、Stop at/inアイコンを選択

.2 プログラムの実行

Runアイコンを選択

.3 ブレークポイントでの停止

.4 プログラムの値の参照

ソースブラウザ中の変数を選択後、Printアイコンを選択

.5 配列の値の参照

ソースブラウザ中の配列を選択後、Graph Matrixアイコンを選択

## 6. チューニングの準備 (Vprof)

.1 性能測定用コンパイルオプションの設定

メニューバーのCompile/ExecからSetCompileOptions.../All Files...を選択し、現れたウィンドウ上でProfilingオプション(Visual Prof:-p)を指定


.2 実行形式ファイルの生成

Buildアイコンの選択... MakeFileの自動作成、コンパイル、リンク

.3 プログラム実行情報の設定

メニューバーのCompile/ExecからSet Run Options...を選択し、現れたウィンドウ上で、実行時に必要な情報を指定。また、性能情報ファイル(a.out.mon.ont)のSXからWSへの自動電送のために、Profile file for Visual Profの項を選択

.4 プログラムの実行

Runアイコンの選択... 性能情報ファイルの生成、およびSXからWSへの性能情報ファイルの自動転送



## 6. チューニングの準備 (PSUITEpa)

### .1 性能測定用コンパイルオプションの設定

メニューバーのCompile/ExecからSet Compile Options.../This File...を選択し、現れたウィンドウ上でProfiling オプション (PSUITEpa[all] or PSUITEpa[routine])を指定。

### .2 実行形式ファイルの生成

Buildアイコンの選択... MakeFileの自動作成、コンパイル、リンク

### .3 性能測定情報の選択

メニューバーのPerformanceからPSUITEpa...を選択し、PSUITEpa起動

#### .3-1. 実行形式ファイルの選択

PSUITEpa Main Window 中の右上ボタンでExecutableが選択されていることを確認 (PDFが選択されているときは、Executableを選択)



#### .3-2. 実行形式ファイルのオープン

PSUITEpa Main Window のメニューバーのFileからOpen Executableを選択し、実行形式ファイルを指定

#### .3-3. 性能測定情報の選択

PSUITEpa Main Window のProfile Selectionを選択し、開いたウィンドウ中の測定したい項目を選択した後、ウィンドウ下部のOKボタンを選択


#### .3-4. 性能測定用実行形式ファイルの作成

PSUITEpa Main WindowのメニューバーのFileからSave Executableを選択

### .4 プログラム実行情報の設定


メニューバーのCompile/ExecからSet Run Options...を選択し。現れたウィンドウ上で、実行時に必要な情報を指定。また、性能報ファイル(a.out.pdf)のSXからWSへの自動転送のために、PDF file for PSUITEpaの項を選択

### .5 プログラムの実行

Runアイコンの選択... 性能情報ファイルの生成、およびSXからWSへの性能情報ファイルの自動転送

## 7. チューニング手順の一例 (Vprof)

### .1 Vprofの起動

Visual Profアイコンの選択

### .2 性能情報ファイルの読み込み

Visual ProfウィンドウのメニューバーのFileからOpen Fileを選択し、現れたウィンドウ上で、Object 文件名 (a.out) とProfile 文件名 (a.out.mon.out) を指定

### .3 性能データの表示

Visual Prof ウィンドウのメニューバーのWindowからSummary, Profile Table(...), Routines 2D Graph(...)などを選択

## 7. チューニング手順の一例 (PSUITEpa)

### 1. PSUITEpaの起動

メニューバーのPerformanceからPSUITEpa...を選択

### 2. 実行形式ファイルの選択

PSUITEpa Main Window中の右上ボタンでPDFが選択されていることを確認  
(Executableが選択されているときは、PDFを選択)

### 3. 実行形式ファイルのオープン

PSUITEpa Main WindowのメニューバーのFileからOpen PDFを選択し、PDF  
ファイルを指定

### 4. 性能データの表示

PSUITEpa Main Window中の下部の項目から、表示したい項目を選択

## 8. 既存のMakefileの利用方法

### 1. プロジェクトへの登録

ProjectメニューからOptions...を選択

利用者のMakefileを使用することを指定

Use User's Makefile を on

利用者のMakefile名を設定

Build用

make -f 利用者のMakefile名

Compile用□ ソースファイル file.f90をコンパイルする場合

make -f 利用者のMakefile名 file.o

### 2. PSUITEの機能を使用するためにMakefileの記述中にコンパイルオプションを追加 最適化ブラウザを使用する場合

-optb

PSUITEpa を使用する場合

-pspa (ルーチンとループの情報を採取する)    o r

-pspar (ルーチンの情報を採取する)

Visual Prof を使用する場合

-p