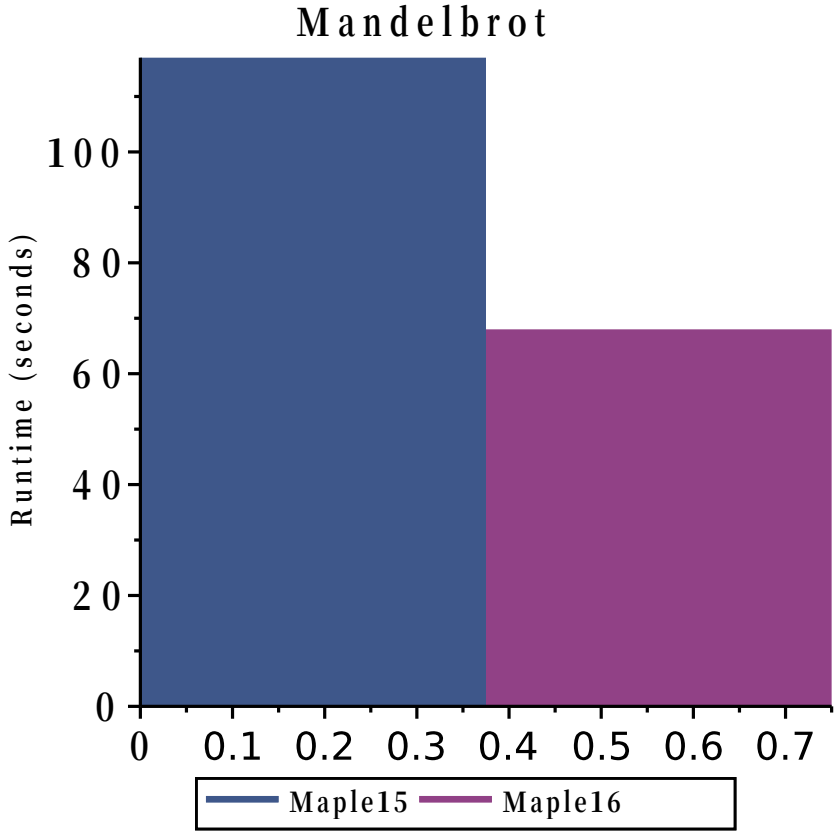
メモリ管理

自動メモリ管理により、プログラマーは計算結果の保存容量の割り当てと開放のための低レベルの記述から開放されます。自動メモリ管理はコストがかかりますが、これによってユーザーを低レベルの作業から開放することで、正確で効率的な高レベルのアルゴリズム開発に集中して問題を解決することで、一般に自動メモリ管理のオーバーヘッドはカバーできます。

Maple のメモリ管理システムを構成するメモリアロケータとガーベイジコレクションに大きな改良が行われました。メモリ管理システムの発展により、現在のマシンのみならず将来のマシンについてもリソースが使用可能になりました。たとえば、CPU 性能の進化によりチップあたりのコア数は増加します。この傾向に応じて、メモリアロケータとガーベイジコレクタの進化は、複数スレッドで同時にメモリを扱うために行われました。さらに、Maple は改良により 64 ビットマシンで大量のメモリを使用できる可能性があります。

最も大きな利点はメモリ管理システムの進化による Maple の並列処理です。従来は、スレッド間の相互作用によりガーベイジが長期間人工的に保たれていました。改良されたガーベイジコレクタは、システム内の特定の Maple スレッドのメモリ使用量を調べ、従来よりも細かい粒度でガーベイジを特定します。このため、大量の計算で必要な時間やメモリ容量が削減されます。これらの改良は、タスクプログラミングモデルのデモで使用されている Mandelbrot の例を参照してください (例、タスク を参照)。クアドコアプロセッサでは割り当てられるメモリ量は約 70% 削減され、パフォーマンスは 42% 改良されています。



従来は、Maple はメモリが必要になったときに、オペレーティングシステムのメモリアロケータから継続的に要求していました。これらのメモリは必ずしも連続したメモリ領域ではなかったため、(他の要求がオペレーティングシステムに送信される代わりに) 大規模のメモリ割り当てを満たすだけの大きさにまとめられることはほとんどありませんでした。多くのプログラミング言語のランタイムシステムで採用されているストラテジを適用することで、Maple は仮想メモリの大規模な連続

ブロックを管理し、必要に応じて拡張することができます。

gc() を呼び出すことによってガーベイジコレクションをトリガすることができますが、推奨しません。Maple のメモリ管理システムは改良により需要主導型になり、gc() を直接呼び出すことによってシステムが実行する作業量が増加する場合があります。

外部ルーチンの実行中のガーベイジコレクションは、歴史的に問題が多いものでした。Maple は、従来は不可能だった多くの場合でのガーベイジコレクションを可能にしました。これによりメモリ使用量とパフォーマンスが改良されました。以下の線形代数の例を考えます。

```
> restart :
  with(LinearAlgebra) :
  Digits := 20 :
   A := RandomMatrix(300, outputoptions = [datatype = float]) :
  B := RandomVector(300, outputoptions = [datatype = float]) :
  try
    X := CodeTools:-Usage( LinearSolve(A, B) ) :
  catch:
  end try:
```

ガーベイジコレクトはこの例では 49% 程度高速化されています。