Chrono & Stat Service*

Vanya Belyaev†
ITEP, Moscow

November 15, 2000

Contents

1 Chrono & Stat Service and *IChronoStatSvc* Interface

Chrono & Stat Service is one of standard components directly visible by Algorithm object. This service is designed to fulfil two different task:

- chrono profiling of the codes ("Chrono" part)
- simple statistical monitoring of useful quantities ("Stat" part)

1.1 Access to the Chrono & Stat Service

Communication with Chrono & Stat Service is performed via an abstract interface *IChronoStatSvc*. Each Algorithm object has the method for accessing to the pointer to an abstract interface of type *IChronoStatSvc*:

```
// The standard Chrono & Stat Service,
// Return a pointer to the service if present
 IChronoStatSvc*   chronoStatService();
```

For accessing the Chrono & Stat Service outside the Algorithm object, one should use standard method of service location, using either the name of the service *"ChronoStatSvc"* or the service ID: *extern const CLID& IID_IChronoStatSvc*.

1.2 Code profiling

Profiling is performed by using *chronoStart* and *chronoStop* methods inside the codes to be profiled, e.g:

```
/// ...
IChronoStatSvc* svc = ...
///  start
svc->startChrono( "Some Tag" );
/// here some user code are placed:
...
/// stop
svc->stopChrono( "SomeTag" );
```

*this document is available in *$CHRONOSTATSVCROOT/svc/ChronoStatSvc/doc* directory
†E-mail:Ivan.Belyaev@itep.ru

The profiling informations accumulates under the tag name, given as an argument for these method. Service perform the measurement of the time between subsequent calls of *startChrono* method and *endChrono* method with the same tag. The later is important, since from the sequence of calls

1. `svc->endChrono("Tag");`
2. `svc->endChrono("Tag");`
3. `svc->startChrono("Tag");`
4. `svc->startChrono("Tag");`
5. `svc->endChrono("Tag");`
6. `svc->endChrono("Tag");`
7. `svc->startChrono("Tag");`
8. `svc->startChrono("Tag");`
9. `svc->endChrono("Tag");`

only the elapsed time between 3 and 5 lines and the elapsed time between 7 and 9 lines would be accumulated.

The information could be printed either directly using *printChrono* method of standard Chrono & Stat Service, or could be printed in the final table of profiling information.

1.3 Statistical monitoring

Statistical monitoring is performed by using the *stat* method inside user code:

```
/// ... Flag and Weight to be accumulated:
svc->stat( " Number of Tracks " , Flag , Weight );
```

The statistical information contains the "accumulated" *flag* , which is the sum of all *Flag*s for the given tag, and the "accumulated" *weight*, which is the product of all *Weight*s for the given tag. The information is printed in the final table of statistics.

In some sense the profiling could be considered as statistical monitoring with variable *Flag* equals to elapsed time of the process.

2 *Chrono* and *Stat* helper classes

To simplify the usage of Chrono & Stat Service, two helper classes were developed: *class Chrono* and *class Stat*. Using these utilities, one hides the communications with Chrono & Stat Service and provides more friendly environment.

2.1 *Chrono*

Chrono is a small helper class, which invokes *startChrono* method of Chrono & Stat Service in the constructor and invokes *endChrono* method in the destructor. It must be used as *automatic local object*. It performs the profiling of the codes since its own creation till the end of the current scope, e.g:

```
/// ...
{ // begin of the scope
   Chrono chrono( svc , "ChronoTag" ) ;
   /// some codes:
    ...
   ///
} // end of the scope
/// ...
```

For usage of *Chrono* utility one must include the file `Gaudi/ChronoStatSvc/Chrono.h`
If the Chrono & Stat Service is not accesible `svc=0`, the *Chrono* object does nothing.

2.2 *Stat*

Stat is a small helper class, which invokes *stat* method of Chrono & Stat Service in the constructor.

```
/// ...
Stat stat( svc , "StatTag" , Flag , Weight ) ;
/// ...
```

For usage of *Stat* utility one must include the file `Gaudi/ChronoStatSvc/Stat.h`
If the Chrono & Stat Service is not accesible `svc=0`, the *Stat* object does nothing.

3 Properties of standard *ChronoStatSvc* service

3.1 Properties for profiling

Profiling properties of the standard Chrono & Stat Service which are configurable via*job options file* and their default values are:

```
// deside if the final printout should be performed;
ChronoStatSvc.ChronoPrintOutTable = true;
// if User   Time information to be printed?;
ChronoStatSvc.PrintUserTime    = true       ;
// if System Time information to be printed?;
ChronoStatSvc.PrintSystemTime  = false      ;
// define the destination of the table to be printed;
ChronoStatSvc.ChronoDestinationCout = false;
// print level for profiling  ( according to MSG::Level)
ChronoStatSvc.ChronoPrintLevel      = 3 ;
// if printout is to be performed,
// should one take care about some ordering?
ChronoStatSvc.ChronoTableToBeOrdered = true ;
```

Tables are ordered according the total elapsed time.

3.2 Properties for statistics

Statistical properties of the standard Chrono & Stat Service which are configurable via*job options file* and their default values are:

```
// deside if the final printout should be performed;
ChronoStatSvc.StatPrintOutTable = true;
// define the destination of the table to be printed;
```

```
ChronoStatSvc.StatDestinationCout = false;
// print level for profiling  ( according to MSG::Level)
ChronoStatSvc.StatPrintLevel      = 3 ;
// if printout is to be performed,
// should one take care about some ordering?
ChronoStatSvc.StatTableToBeOrdered = true ;
```

4 Performance consideration

Implementation of Chrono & Stat Service used two *std::map* containers. Therefore for very frequent calls it could cause the performance penalty. Usually this penalty is negligible with respect to the elapsed time of algorithms, but it is worth to avoid both the direct usage of Chrono & Stat Service as well as the usage of it through *Chrono* or *Stat* utilities inside the internal loops:

```
/// ...
{  /// begin of the scope
Chrono chrono( svc , "Good Chrono");  /// OK
long double a = 0 ;
for(  long i = 0 ; i < 1000000 ; ++i )
 {
 Chrono chrono( svc , "Bad Chrono");   /// not OK
 /// some codes :
   a += sin( cos( sin( cos( (long double) i ) ) ) );
 ///  end of codes
Stat   stat  ( svc , "Bad Stat", a ); /// not OK
 }
Stat   stat  ( svc , "Good Stat", a); /// OK
} ///  end of the scope!
/// ...
```