The following table lists GEOPM markup routines, description and the corresponding attributes in Caliper to be reported to GEOPM by the proposed GEOPM service in Caliper.

GEOPM markup	Description	Attributes/Callbacks

`int geopm_prof_region(`
`  const char`
`  *`*region_name*,
`  long` *policy_hint*,
`  uint64_t *`*region_id*)
	Registers an application region. The *region_name* and *policy_hint* parameters are input parameters, and the *region_id* is output.
	(Note: GEOPM doest not support nested regions as of now, so only return the top-level region ID)
		Caliper: region ID
		Unresolved: policy hint
		Scope: process local
		Callback: pre_begin_evt

`int geopm_prof_enter(`
`  uint64_t` *region_id*)
 	
`int geopm_prof_exit(`
`  uint64_t` *region_id*)
	Called by the compute application to mark begin and end of the profiled compute region associated with the *region_id*.	Caliper: region ID
		Scope: process local
		Callback: pre_begin_evt, pre_end_evt

`int`
`geopm_prof_progress(`
`  uint64_t` *region_id*,
`  double` *fraction*)
	Called by compute application in single threaded context to signal the fractional progress, *fraction* through the work required to complete the region where *fraction* is between 0 and 1.	Caliper: region ID
		Scope: process local-only
		Callback: pre_end_evt
		Additional call required to report: 'fraction'

`int geopm_prof_epoch(` *void)*	Called once for each pass through a computational loop containing inter-node synchronization events. Acts as a beacon signal emitted by each MPI rank as it begins a loop iteration.	Attribute: Scope: process local Callback: pre_end_evt This is mapped to a the end of region ID named 'mainloop' to indicate end of the main loop as required by GEOPM.

`int geopm_prof_disable(`
`  const char`
`  *`*feature_name*)
	Called at application start up to disable a profiling feature.	Scope: process local
		Not implemented in GEOPM.

`int geopm_tprof_init (`
`  uint32_t` *num_work_unit*)
 	
`int`
`geopm_tprof_init_loop (`
`  int` *num_thread*,
`  int` *thread_idx*,
`  size_t` *num_iter*,
`  size_t` *chunk_size*)
	Create a thread profiling object, *tprof*, which extends the functionality of the profiling interface to report progress within threaded regions.
	
	Assume a fixed number of threads, *num_thread*, which are performing work sharing on a list of tasks *num_iter* long (e.g. an OMP parallel for loop with *num_iter* loops).
		Attribute: Annotation, loop
		Scope: thread local
		Callback: pre_end_evt

`int geopm_tprof_post (` *void*)	Called after a thread has completed each work unit to report progress.	Attribute: Annotation, loop Scope: thread local-only Callback: pre_end_evt Must not be called along with geopm_prof_progress( )