

polly.task_entry:

```
%polly.pthread_args = bitcast i8* %task_args to { i64, i32, i64, double*,  
... double*, double* }*  
%0 = getelementptr inbounds { i64, i32, i64, double*, double*, double* }, {  
... i64, i32, i64, double*, double*, double* }* %polly.pthread_args, i32 0, i32 0  
%threadId = load i64, i64* %0  
%1 = getelementptr inbounds { i64, i32, i64, double*, double*, double* }, {  
... i64, i32, i64, double*, double*, double* }* %polly.pthread_args, i32 0, i32 1  
%polly.taskfn.arg.n = load i32, i32* %1  
%2 = getelementptr inbounds { i64, i32, i64, double*, double*, double* }, {  
... i64, i32, i64, double*, double*, double* }* %polly.pthread_args, i32 0, i32 2  
%polly.taskfn.arg. = load i64, i64* %2  
%3 = getelementptr inbounds { i64, i32, i64, double*, double*, double* }, {  
... i64, i32, i64, double*, double*, double* }* %polly.pthread_args, i32 0, i32 3  
%polly.taskfn.arg.x1 = load double*, double** %3  
%4 = getelementptr inbounds { i64, i32, i64, double*, double*, double* }, {  
... i64, i32, i64, double*, double*, double* }* %polly.pthread_args, i32 0, i32 4  
%polly.taskfn.arg.A = load double*, double** %4  
%5 = getelementptr inbounds { i64, i32, i64, double*, double*, double* }, {  
... i64, i32, i64, double*, double*, double* }* %polly.pthread_args, i32 0, i32 5  
%polly.taskfn.arg.y_1 = load double*, double** %5  
br label %polly.parallel.for
```

polly.parallel.for:

```
%6 = sext i32 %polly.taskfn.arg.n to i64  
%7 = sub nsw i64 %6, 1  
%polly.fdiv_q.shr = ashr i64 %7, 5  
%polly.par.userContext = bitcast { i64, i32, i64, double*, double*, double*  
... }* %polly.pthread_args to i8*  
%8 = add i64 %polly.fdiv_q.shr, 1  
call void @GOMP_parallel_loop_runtime_start(void (i8*)*  
... @polly_task_polly_subfn, i8* %polly.par.userContext, i32 0, i64 0, i64 %8,  
... i64 1)  
call void @polly_task_polly_subfn(i8* %polly.par.userContext)  
call void @GOMP_parallel_end()  
br label %polly.task_exit
```

polly.task_exit:

```
call void @pthread_exit(i8* null)  
unreachable
```

CFG for 'polly.task' function