Ticket #101

- Communicator Hints for upscaled reduction ops
- Explicit assumption in reduction ops
  - The intermediate computation will be carried out by using the same datatype
- Indeed, especially in FP16, rounding error can be a severe problem
- Hint to have upscaled intermediate computation in reduction
Upscaling Hints (1/3)

- Tradeoff between rounding error and speed (+memory)
- Minimize accuracy differences among implementations
- Upscaling can not be avoided on some H/W environments (contradicts with upper)
  - E.g., NIC supports FP32 and FP64 (no FP16)
- One(-level) upscaling hint may not be enough
  - Global FP16 sum on millions of ranks
  - Two(-level) upscaling hints
- Implementation might be hard
  - to have all possible upscaling combinations
Up-scaling Hints (2/3)

```
TYPE_ORG   *sendbuf, *recvbuf;
TYPE_UPSscaled *sb_upscaled, *rb_upscaled;

UPSCALING( sb_upscaled, sendbuf, count );  // vectorizable
MPI_Reduce( sb_upscaled, rb_upscaled, count, UPSCALED, SUM, ... );
DOWNSCALING( recvbuf, rb_upscaled, count ); // vectorizable
```

- Automatic upscaling vs. explicit user-up-scaling
- How slow is user-scaling?
  - Communication time: up-scaling + down-scaling
### Upscaling Hints (3/3)

```c
TYPE_ORG   *sendbuf, *recvbuf;
MPI_Reduce(sendbuf, recvbuf, count, datatype, SUM, ... );

TYPE_ORG   *sendbuf, *recvbuf;
TYPE_UPSACLED  sum;
for( i=0; i<count; i++ ) sum += (TYPE_UPSACLED) *(recvbuf++);
*recvbuf = (TYPE_ORG) sum;

FP16      x, y;
y = expS( y );  // y = (FP16) expL( (double) x );
```

- **Pt2Pt2 WG on Wednesday**
  - **NO:** 4/10
  - **YES:** 0/10
  - **ABSTAIN:** 0/10