As for the suitability of ROMS as oceanic component for climate modeling, there is no simple answer: on one hand it is "why not?" simply because the dynamical core of ROMS code (basis algorithms such as time stepping, mode splitting, advection for momentum and tracers), etc are already in place along with necessary peripheral components KPP (lateral mixing, etc...), and whatever is missing can be added, brought in, ported, etc...

On the other hand, ROMS was never used for anything close to climate modeling (besides running Pacific model for 30 and 50 years), so in this sense it is kind of "unproven". It is however very proven in business of high-resolution highly turbulent flow regimes.

My current kernel design -- a tightly integrated code structure ("anti-modular" by its philosophy) incorporating everything what has been done in 15 years, from basic time stepping algorithms, splitting/coupling to adaptively implicit vertical advection synergetically coexisting with iso-something rotated (isoneutral or whatever one wishes to rotate it to) [hyper-]diffusion stabilized by partial vertical implicitness, and all the way to revised mode splitting (yet again), all embedded into parallel shell capable of 2-level domain decomposition (sub-tiling inside MPI) -- actually makes me proud.

How do we proceed from where we are right now?

It should begin with an inspection of what we have and what is needed. I kind of expect a set of evaluation tests, perhaps something in spirit of CORE 2 and see how it goes. Obviously bringing some components ROMS currently lacks. I do not expect it quick and easy -- I generally do not like bringing pieces of codes from outside and implanting them into ROMS code by copy-paste approach (see below), but I do not see any fundamental obstacle in carefully revising and reworking such pieces whenever necessary.

Not long ago I worked on isoneutral diffusion and bulk fluxes. Both are functional now in UCLA branch, but I feel more can be done.

I plan to work on ice model in near future, but to say it realistically, what is in ROMS right now about ice is out of date, given the developments in CICE and LIM3 and overall in ice community.

Target resolutions: do you consider 0.25 degree (or finer) as a practically realistic oceanic model for climate from the point of view of computational resources?

Another natural thought is that, as I understand, the future WRF model will be MPAS. Which means that it would be logical to have an MPAS-like (unstructured) oceanic component. ROMS is a structured grid model. Los Alamos people work on MPAS oceanic code, which makes it a natural choice for partner, but... Do you consider any other MPAS-like oceanic model?

For basin-scale ROMS modeling, and perhaps for global, I kind of see the use of general curvilinear grids with combination of land masking to make variable resolution with focus on specific places. I have seen several papers which look promising, but my own experience in generating such grids is very limited at this point. ROMS code itself is written for general orthogonal grids, retaining all the metric terms, so it is kind of ready for that.

I am obviously hesitant for another good reason: over very long time I was working predominantly on the hydrodynamic kernel and code architecture covering all the issues involved, from numerical algorithms to optimization of the code, parallel performance, and even working below the ground: building computers, operating systems, disk storage systems, and worrying about performance. This

makes me essentially work alone -- 100% lonely, but got used to it and do not complain: nobody wants to do it any way, but also gives an opportunity to test ideas which would be impossible to do is working in an institution, or, realistically speaking, even in a team. As the result, I generally keep low profile, in fact, very low during the last several years. Certainly some may say that this is a wrong attitude, asocial, etc.. -- but who among those saying ever worried about CAS latencies or ratio of inner-level-cache-to-register memory bandwidth vs. that of main-memory-to-CPU and how it practically translates on Fortran code? In fact, most of the time people want to see and discuss the bulk result -- a working code -- not about what is underneath and why. To a significant degree my behavior was formed by two coincidences: (1) at one time I happened to closely know an SGI insider Wesley Jones at the time when SGI was hot, and (2) in fall 1995 Jim sent me to a supercomputing workshop hosted by Paul Woodward. In retrospect I would have to say that Woodward's group was way ahead of the time back then in 1995 in sense of anticipating direction to which computer architectures go and, most importantly, knowing what to do about it in terms of practical design of fluid-dynamics codes. This ultimately set me in a collision course with a sequence of matlab-loving postdocs here in UCLA, who basically say lets don't worry about all your experiences, let's do science first: we will fix codes later if time allows. A politically correct and socially comfortable line of behavior from my side would be "to be like everybody around", but this bores me to death: not because I like arguing for the sake of arguing (in fact I usually avoid) but because I am getting bored every time when I see a 6-core CPUs wasted by running for several hours somebody's Python (or Matlab) script utilizing only a small fraction of a single core performance out of six (small fraction because such scripts are unavoidably memory bandwidth bound and at best the utilize BLAS routines, but any algorithm slightly out of ordinary would not be even pipelined). Bored because I believe that there is a better way, yet all what I hear is "but it does the job, so what? Ok, if you can do it better, please provide it for us and we promise to use it, but until then we will use what we have".

And here comes the problem: Whenever I see an institutionalized oceanic code -- MOM4p1, NEMO, POP, HYCOM, GOLD, etc (to a much lesser degree MITgcm) -- I do not like what I see. I know that what I am saying might sound harsh, but this is what it is (particular examples with my commentary are available upon request if someone asks): all these codes bear scars of two-level development approach with one-way-communication: there are "physical scientists" who just tell what to do to programmers (engineers, computer scientists, whatever you call them) who are just doing their job. The outcome is a bulky code written without an intent to be understood, usually without meaningful semantic rules, not commented at all (corporate logos, copyright notices, comment lines duplicating Fortran subroutine names, and straight horizontal dashed lines do not count). Existence of official documents and guidelines -- such as "European Standards for Writing and Documenting Exchangeable Fortran 90 Code" or "FORTRAN Coding Standard in OPA System - Nemo" or even "The Parallel Ocean Program (POP) Reference Manual" of 2004 (for POP v2.1) and its 2010 update -- are mostly contributions to confusion rather than practical examples to follow: formally these documents mention all the relevant subjects and superficially looks credible: 99% of what is written there is hard to object. POP v2.1 manual even mentions importance of code optimization for reuse of cached data, and even mentions sub-tiling. But how does this reflect in the code itself? Not much at all. Basic neglect of the fact that division operation is several times more expensive that multiplication. Neglect of the notion that loops need to be of a certain optimal complexity in order to be efficiently pipelined; the fact that if-statements do not avoid operations inside a pipelined loop. Lack of meaningful policy for loop boundaries associated with 2D domain decomposition (tiled) parallel code. Typically lack of meaningful scratch memory management -- which of the codes above require unlimited stack size or -heap-arrays compiler flag to run? All? If a well motivated and knowledgeable person wants to bring a component, say a biological or a ice model model he must "play by the rules", which means that the outcome will be ...as usual.

I believe there must be a better way.