new - Java, Ruby, Scala, Go
old - DCE-RPC, CORBA, DCOM
RPC rename peer calls
Client Store

Secure Store

Local Comprehend

Strategy

If (user is legit) =流行旧

IDL write the interface
I learned something new today.

In the class, a senior speaker shared the importance of critical thinking and problem-solving. Emerson, Chekhov, Proust, and Mendelssohn.
[Source] a unique mesh - [un]

Client

foo(x)

message
Client stub

send (sp, msg) => sendEvent without error

} catch (err) {
  ITT
}

return;
DSM distributed shared memory

coherent shared memory on NORMA system
Shared Segment: Plan as 5

Globally,

Process:

\[ x = 5 \]
client page fault handling

1. Read p.f

2. Send request to DSM server with p#, r mode

   - get contents of page
   - put new contents into mem & set prot to R-only

}
Set permissions to RW
get path
read scenarios for path in Luke mode
 Ask scenarios for path in Luke mode (current path mode)
Every set \( S \) with \( \lambda \) is isomorphic to \( \mathbb{R} \) in \( \omega \), \( \langle \omega, \leq \rangle \). Every pre-order is well-founded for \( \lambda \). Every class \( A \) is a copy of \( \mathbb{R} \) and \( \lambda \) is a copy of \( \omega \). Every \( \mathcal{L} \) is an initial segment.
none

\( \sqrt{\text{mode}} \)
Set to "new"

All mode

Send back to F

Set mode to M

Work for all receivers (0, 0, 0)

All processes in Log

Send initialization message to

"part in A mode"

"serve get W request, Fram"
- Send packet to P1
- Set mode to U1, cañón = P2
- Get packet back, rep capăy in main
- Send w-inoculadora to P1
- Part in w mode (coáñer = P1)
- Server gets w request by P1
Send copy of pose
set mode to move
  M - Invertible
  R - Inverted
  R - Inversed

  set mode to move
  set mode to move

  - Hpem

  - Hpem

  - Hpem
Look up local maps? Is not clear here?
\[ x = \frac{1}{x+1} \]

\[ x + x = \frac{1}{x+1} \]

\[ x = \frac{1}{x+1} \]

Diagram:
- Circle labeled '+'
- Arrow pointing to 'x'
- Arrow pointing to '1'
- Arrow pointing to 'x+1'
- Arrow pointing to 'x'
x and y on some part

false sharing

y++

x++
Release consistent DSM
- involves loading into DSM server
- allows multiple W copies
- uses merging via XOR
- no page thrashing (false sharing)